NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

HYBRID HIGH-FIDELITY MODELING OF RADAR SCENARIOS USING ATEMPORAL, DISCRETE-EVENT, AND TIME-STEP SIMULATION

by

Yuan-Pin Cheng

December 2016

Dissertation Supervisors: Donald P. Brutzman
Phillip E. Pace

This dissertation was performed at the MOVES Institute.
Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 2016	Dissertation 12-27-2012 to 12-16-2016

4. TITLE AND SUBTITLE

HYBRID HIGH-FIDELITY MODELING OF RADAR SCENARIOS USING ATEMPORAL, DISCRETE-EVENT, AND TIME-STEP SIMULATION

5. FUNDING NUMBERS

6. AUTHOR

Yuan-Pin Cheng

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

MOVES Institute, Center for Joint Services Electronic Warfare
Naval Postgraduate School Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

Many simulation scenarios attempt to seek a balance between model fidelity and computational efficiency. Unfortunately, scenario realism and model level of detail are often reduced or eliminated due to the complexity of experimental design and corresponding limitations of computational power. Such simplifications can produce misleading results. For example, Radar Cross Section (RCS) effects in response to time-varying target aspect angle are dominant yet often ignored. This work investigates a new approach. A hybrid, high-fidelity sensor model can be achieved by using a Time-Step (TS) approach with precomputed atemporal response factors (such as RCS), each situated on active simulation entities that interact within an overall Discrete Event Simulation (DES) framework. This work further applies regression analysis to the cumulative results of 1,281 design points multiplied by 100 replications each to provide additional insight from massive results. Through a rigorous cross-validation (CV) for ensuring proper model selection, this new methodology adapts the best aspects of temporal control by each simulation approach, integrating multiple high-fidelity physically based models in a variety of tactical scenarios with tractable computational complexity. The meta-model generates statistically identical results as the baseline hybrid sensor model in mean with higher variation. This outcome sensor model performs in an efficient DES architecture by predicting only probability of detection (Pd) and time-to-detect (TTD) of the targets with the consideration of the range and the aspect relationship between the sensor and the target. The range and the aspect factors were ignored in current existing DES sensor models due to the overly simplified detection algorithm and the constraint of modeling complexity.

14. SUBJECT TERMS

Discrete Even Simulation (DES), radar range equation, Simkit, Visual Simkit, atemporal model, hybrid sensor model, DES event graph, Nearly Orthogonal Latin Hyper Cube (NOLH), regression analysis, cross-validation

15. NUMBER OF PAGES 223

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

HYBRID HIGH-FIDELITY MODELING OF RADAR SCENARIOS USING ATEMPORAL, DISCRETE-EVENT, AND TIME-STEP SIMULATION

Yuan-Pin Cheng
Major, The Republic of China, Taiwan Army
B.S., National Defense University, R.O.C., Taiwan, 2000
M.S.E.E., Naval Postgraduate School, 2008

Submitted in partial fulfillment of the
requirements for the degree of

**DOCTOR OF PHILOSOPHY IN
MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION**

from the

**NAVAL POSTGRADUATE SCHOOL
December 2016**

Approved by:

Donald P. Brutzman
Associate Professor of
Applied Science, MOVES
Dissertation Co-Supervisor

Phillip E. Pace
Professor of Electrical and
Computer Engineering and
Director, Center for Joint
Services Electronic Warfare
Dissertation Co-Supervisor

Arnold H. Buss
Research Professor of
Operations Research, MOVES

David C. Jenn
Professor of Electrical and
Computer Engineering

Thomas W. Lucas
Professor of Operations
Research, MOVES

Approved by: Christian J. Darken, Chair, MOVES Academic Committee

Approved by: Peter J. Denning, Chair, Department of Computer Science

Approved by: Douglas Moses, Vice Provost of Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Many simulation scenarios attempt to seek a balance between model fidelity and computational efficiency. Unfortunately, scenario realism and model level of detail are often reduced or eliminated due to the complexity of experimental design and corresponding limitations of computational power. Such simplifications can produce misleading results. For example, Radar Cross Section (RCS) effects in response to time-varying target aspect angle are dominant yet often ignored. This work investigates a new approach. A hybrid, high-fidelity sensor model can be achieved by using a Time-Step (TS) approach with precomputed atemporal response factors (such as RCS), each situated on active simulation entities that interact within an overall Discrete Event Simulation (DES) framework. This work further applies regression analysis to the cumulative results of 1,281 design points multiplied by 100 replications each to provide additional insight from massive results. Through a rigorous cross-validation (CV) for ensuring proper model selection, this new methodology adapts the best aspects of temporal control by each simulation approach, integrating multiple high-fidelity physically based models in a variety of tactical scenarios with tractable computational complexity. The meta-model generates statistically identical results as the baseline hybrid sensor model in mean with higher variation. This outcome sensor model performs in an efficient DES architecture by predicting only probability of detection (Pd) and time-to-detect (TTD) of the targets with the consideration of the range and the aspect relationship between the sensor and the target. The range and the aspect factors were ignored in current existing DES sensor models due to the overly simplified detection algorithm and the constraint of modeling complexity.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

2D	Two dimensional
3D	Three dimensional
ADC	Analog-to-Digital Converter
AIC	Akaike Information Criterion
AF	Attenuation Force
BIC	Bayes Information Criterion
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CI	Confidence Interval
CPA	Closest Point of Approach
CV	Cross Validation
DES	Discrete-Event Simulation
DoE	Design of Experiment
DP	Design Point
HPC	High Performance Computing
LEGO	Listener Event Graph Object
M&S	Modeling and Simulation
MSE	Mean Squared Error
MTTD	Mean Time To Detect
NOLH	Nearly Orthogonal Latin Hypercube

NPS Naval Postgraduate School

NVE Networked Virtual Environment

OMG Object Management Group

P_d Probability of Detection

PDF Probability Density Function

PEC Perfect Electric Conductor

PK Probability of Kill

prf pulse repetition frequency

pri pulse repetition interval

RASE Root Average Squared Error

RCS Radar Cross Section

RF Radio Frequency

Rmax maximum detection range

RMSE square Root of the Mean Squared prediction Error

rpm revolutions per minute

RRE Radar Range Equation

RSS Residual Sum of Squares

Ru maximum Unambiguous range

RV Random Variable

SAM Surface-to-Air Missile

SFD Space-Filling Design

SNR Signal-to-Noise-Ratio

SSE	Sum of Squared Errors
STFT	Short-Time Fourier Transform
ToT	Time on Target
TS	Time Step
TSS	Total Sum of Squares
TTD	Time to Detect
UAV	Unmanned Aerial Vehicle
UML	Unified Modeling Language
Viskit	Visual Simkit
VRML	Virtual Reality Modeling Language Standard
VV&A	Verification, Validation and Accreditation
WT	Wavelet Transfer
X3D	Extensible Three-Dimensional Graphics Standard

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

In a dynamic sensor-targets simulation scenario, the location and aspect angle between the sensor and the targets vary over time. The range and the angle play a key role in the sensor's detection ability based on the radar range equation (RRE) and aspect-dependent signal response.

As more detailed parameters are integrated into the system for a higher-fidelity modeling result, simulation practitioners always face "triangle of death" tradeoffs in pursuing the optimal balance between "faster, better and cheaper." A balance must be found because there is no single perfect solutions for the model. However, a modeling architecture that integrates high-fidelity physics-based parameters with coherent response times for all factors is crucial for a simulation to perform coherently and accurately.

This work demonstrates a hybrid-structure sensor model that successfully integrates high-fidelity RRE with dynamic radar cross section (RCS) response from varying-aspect targets in a scenario of a stationary ground-based searching radar system. Based on computationally efficient discrete event simulation (DES) structure, the simulation isolates insignificant events out of computational consideration by utilizing the maximum unambiguous range (Ru) of the sensor as the high-fidelity event trigger boundary. When the target movement is out of Ru, the interaction events to the sensor model can be omitted since these is no detection event for the sensor in that range. All events are scheduled precisely at the moment of event-state transition. This event-driven architecture leads to no ambiguity on simulation schedule timing and much great fidelity within longer simulations.

The sensor model isolates a high-complexity atemporal RCS model and integrates a time-step (TS) mechanism for emulating the antenna sweeping behavior. The target RCS model is laboriously calculated in advanced by the electron magnetic simulation software, CST Studio, with the 3D object models in a real size scale. With the advantage of cluster based computational power and atemporal model classification, the RCS computational latency can be shortened and isolated from the whole system response time. Dominant RCS effects are considered in the function of aspect angle of the target for satisfying a dynamic sensor-target simulation scenario. The time-step antenna scanning behavior is

scheduled repeatedly using precise time steps in the interval of the antenna revolution period under the DES architecture.

The RRE is applied for determining the maximum detection range of the target to the sensor by the current geometry between two objects at each scheduled detection attempt event. With consideration of the target RCS fluctuation loss, the Swerling case III model is introduced into the RRE and makes the simulation as a stochastic system. With the flexible and efficient hybrid sensor model architecture, the target entities are extended into eight different aircraft models for a further large scale simulation application.

This hybrid model not only introduces the angle-varying RCS factor and high-fidelity radar parameters into the dynamic model scenario, but also keeps the computational complexity of the whole scenario down to a reasonable level. A typical desktop PC is capable of executing 1,281 design of experiments (DoE) with 100 repetitions in a matter of minutes. Even with these benefits of a DES-based structure for the hybrid model, the embedded TS mechanism for emulating the antenna sweeping period is still a key latency for a comprehensive model. Additionally, the model also needs to include a maximum detectable range table for the target. These factors still make the model bulky in some sense, but further improvements are now feasible. This research also utilizes efficient DoEs by Nearly Orthogonal Latin Hypercube (NOLH) design, which provides a well-filled space of parameters with significantly fewer design points compared with an exhaustive full-factorial design. This efficient DoE comprehensively tests the hybrid sensor model behavior for all parameters across a satisfactory range of simulation scenario attempted. Not only are the high-level resolutions covered in the range of parameters, but the interaction relationships between each parameter are also satisfactorily illustrated by the DoEs.

Finally, regression analysis is applied based on the observations executed by the DoEs for studying the significant factors of the hybrid sensor model. A logistic-regression meta-model and a linear-regression meta-model are performed respectively for predicting the probability of detection (Pd) and the time-to-detect (TTD) the target by the sensor. The meta-models do present the hybrid model characteristics in a certain sense with much more concise structure, even if the meta-models are curve-fit approximations that show some heteroscedasticity in prediction results.

Interestingly, with the cross-validation (CV) performed by a generous amount of randomly

selected independent datasets, these meta-models show a low estimation error with no significant sign of over-fitting. The regression approximation analysis for the simplifying the hybrid sensor model execution complexity and usability to a certain degree.

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

This dissertation is dedicated to my beloved wife, WanJen, for she has always been the solid support for me during these four years. I gratefully appreciate her enormous sacrifices in taking care of our whole family while I buried myself in the research. To my two adorable kiddos, Yvonne and Shawn, thank you for giving me nothing but joy while I have been overwhelmed by the data and equations. Also to my dear parents who endured loneliness and separation from us, but they still continued supporting me spiritually.

I believe that I have the best and most knowledgeable committee that any student could ever have. I would like to gratefully acknowledge my two supervisors, Dr. Don Brutzman and Dr. Phillip E. Pace, for their extraordinary guidance and patience so I could have courage and direction to explore a new simulation architecture between electronic engineering and virtual environment simulation fields, which has rarely been accomplished before. The result of this research was enlightening, and all the effort was worth it.

I am sincerely grateful to Dr. Arnold Buss for educating me about the DES concept, which is one of the main modeling architectures in the research. He also provided much support in accomplishing the hybrid sensor structure implementation on Simkit.

I would like to thank Dr. David C. Jenn for the guidance on radar system parameters calibration and for insight and support on the radar cross section computation in CST Studio. My gratitude goes to Robert Broadston and Paul Buczynski for the laboratory assistance.

I must also thank Dr. Thomas W. Lucas and the Simulation Experiments Efficient Designs (SEED) Center of NPS for providing guidance on the design of experiment and simulation analysis experience. My appreciation also goes to Mary L. McDonald, who gave me guidance on using the data analysis application created by JMP Pro software.

A special thank you is owed to NPS Hamming High Performance Computing (HPC) cluster and Brian Andrus for assistance establishing the simulation environment on the cluster. With the tremendous computational power support from the cluster, I was able to bring nearly unavailable and highly detailed RCS from the arbitrary models for use as tractable

data in the simulation.

Also I would like to express my thanks to Terry Norbraten, Jimmy Liberato and all the faculty in the MOVES institute. Without all of your support, guidance, and sharing of knowledge in a variety of perspectives, this work would not have been completed and fruitful.

This study is supported in part by the Office of Naval Research (ONR) in Arlington, VA, for experimental hardware and article publication. I also proudly dedicate my contribution to my beloved country, the Republic of China, Taiwan, for sponsoring me to pursue this ultimate academic degree and knowledge.

Last but not least, God has given me the gifts and strength that have made me successful. Without leading from the LORD, I would forever be lost. *" I am the vine; you are the branches. If you remain in me and I in you, you will bear much fruit; apart from me you can do nothing."* - John 15:5

CHAPTER 1:
Introduction

There are two kinds of time-steps: too small and too big.

–Arnold H. Buss [1]

1.1 Background

Radar systems are a crucial information-acquisition feature in Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR), especially in military tactical applications. The battlefield environments inherently contains a lot of variation and unknown features due to varied types of weapon platforms, such as system arrangements, environmental barriers, tactical decisions [2]. It can also be time consuming, costly, or even infeasible to arrange or reproduce scenarios through physical hardware devices.

A radar system simulation is an essential capability for decision makers or weapon system operators who need to evaluate and predict possible scenario consequences beforehand. Radar systems are a sophisticated combination of hardware, software, and data systems with a variety of features, functions, and configurations. Radar signal-propagation software is also a complex nonlinear physical model that associates with many factors in the environment. Dynamic Three dimensional (3D) visualization can significantly improve researchers' and operators' understanding of these invisible signals. 3D objects often provide a more intuitive approach for understanding the geometry of the object.

Simulation models are increasingly being used to solve problems and to aid in decision making. The decision makers use information obtained from the results of the models in simulation scenarios. Therefore, decision accuracy is highly dependent on whether the models and the results used for decision reference are "correct" [3]. From a simulation perspective, practitioners must always consider data Verification, Validation and Accreditation (VV&A) throughout the models that are considered. The process of verifying the internal consistency and correctness of data, validating that it represents real-world entities appropriate for its intended purpose or an expected range of purposes,

and accreditation is the official certification that a Modeling and Simulation (M&S) is acceptable for use for a specific application [4]. In order to produce a validating model that can represent a real-world scenario, the model construction and responding behavior have to be as close as possible to the object that the simulation intends to represent. Therefore, the implementation of both model and scenario need to be based on the true physics algorithm for having genuine behavior.

1.2 Motivating Problem

Temporal modeling always presents a dilemma for simulation developers and users. On the one hand, modelers want the simulation to contain as many details as possible, so that results might be closer to the real world. On the other hand, simulation computation needs to be executed efficiently and smoothly. As the models get more complicated, a computational complexity and cumbersome system response are the price paid, especially when dealing with a high-complexity physical model, like a radar system, in simulation. Better levels of detail create higher computational complexity, which negatively impacts efficiency. Such degradation can limit the practical usefulness of a physics-based simulation.

A model should be developed for a specific purpose and its validity determined with respect to that purpose. It is often too costly and time consuming to determine that a model is absolutely valid over its full range of intended application. Figure 1.1 illustrates the relationship between the model confidence (also known as the model validity) and the cost of model validation. As the model confidence increases to a certain level, the cost of such a high confidence model has increased by exponential increments, while the value of the model to the user does not have an equally likely positive response. This explains why an extremely high model confidence does not always guarantee that the model is valid everywhere in its application domain.

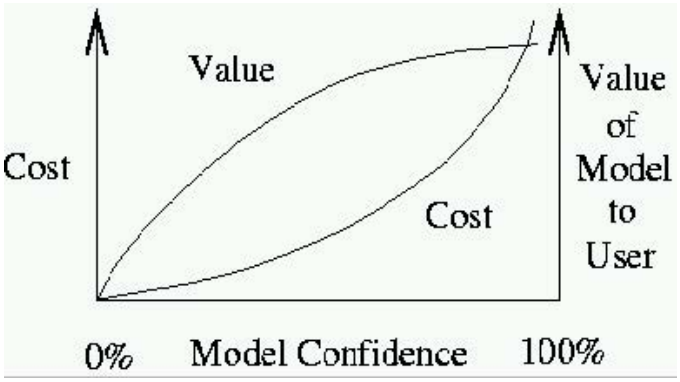


Figure 1.1. Model confidence vs. model cost. Source: [3].

Time Step (TS) or fixed step architectures are widely used in the engineering application field and the granularity of the step size play a key roll of the system performance. Powers et al. [5] study the performance of signal sampling by a wide-bandwidth opto-electronic Analog-to-Digital Converter (ADC). Figure 1.2 shows a sampling optoelectronic architecture. The Radio Frequency (RF) signal is sampled by an optical pulse train in an optical modular. The RF amplitude modulates the laser pulse and the optical laser pulses are detected and electronically digitized. For no aliasing signal sampling result, the sampling frequency has to be at least twice the highest frequency of the signal or more, also known as Nyquist rate. in considering of pulse width, the timing jitter, and the amplitude jitter of the sampling pulse train, the signal pulse width, $\triangle t$, must meet the condition

$$\triangle t \leq \frac{\sqrt{\frac{3}{2^{n-1}}}}{\pi f_m} \tag{1.1}$$

where n is the desired number of bits and f_m is the maximum frequency in the signal. The pulse width is inverse-proportional to the maximum signal frequency. In other words, the higher the signal frequency, the narrower and pulse width will be. The benefit of using optoelectronic ADC is that the high frequency optical signal can sampling the signal with much wider bandwidth then the lower sampling frequency electronic ADCs. The 10-bit electronic ADC can only sample the signal bandwidth approximately up to 200 to 300 MHz while the an opto-electronic ADC would represent a significant achievement in ADC performance to 10-GHz signal bandwidth.

A high detailed fixed step simulation for analyzing the performance of photonic sigma-delta

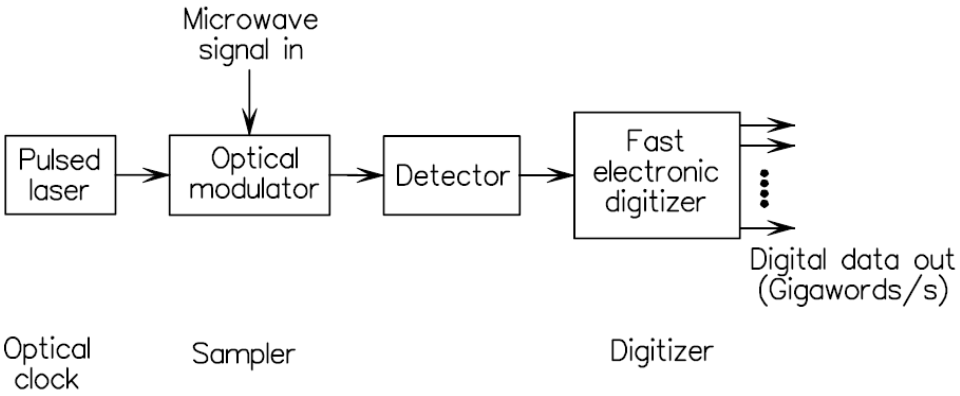


Figure 1.2. ADC sampling-architecture. Source: [5].

modulator is proposed by Tan et al. [6]. A sigma-delta modulator is an oversampling architecture that consists of an analog noise-shaping filter enclosed within a feedback loop around a quantizer. Since the signal is oversampled at many times the Nyquist rate, the high oversampling rate removes the high frequency quantization noise, down samples the modulator output to Nyquistand, and increases the resolution of the output digital signal. The electronic converters, however, have traditionally been limited to low and moderate signal frequencies due to the much lower sampling rate than to photonic modulators. A software simulation structure follow closely that of the physical hardware is proposed in the study. Because the oversampling frequency by photonic sigma-delta modulator is high, this high sampling rate brings the extremely computational complexity to the simulation architecture. The system parameters such as the laser frequency, the laser pulse width, the laser pulse repetition frequency (prf) must be scaled down due to the limited memory space and processing time. This is an illustration of modeling cost increment associating with the modeling fidelity.

Cheng [7] proposes a TS-based simulation algorithm for extracting time information of a frequency-hopping signal by temporal-correlation functions, wavelet transformation, and image recognition schemes. The study shows the capability of TS structure to extract the time-varying information out of the sampling data in different TS resolutions. The left plot of Figure 1.3 illustrates a Short-Time Fourier Transform (STFT) method that applies a fixed TS sample size window to sample a time-series signal that potentially contains different signal frequencies over different times. The fixed-width sample window slides through the signal along the time axis in the same direction as the blue arrow shown in the left plot.

The right plot is the result of the STFT that converts the signal from the time domain into the frequency domain in a fixed δt data-sampling resolution, i.e., the width of the sample widow. The blue cell filled in the time-frequency contour plot indicates the present time and frequency of the signal, which is sampled and converted by the STFT at that specific sampling time (left plot). Because the duration of signal frequencies in a frequency-varying signal is different from hop to hop, the frequency-hopping state transition happens in a different time interval. A fixed TS of data sampling demonstrates the typical challenge of choosing the proper TS scale for the good enough resolution for capturing the interested information. For the too-coarse TS sampling interval, it may lose the precision of time information when the frequency state transition happens between a sample window. For the too-fine TS sampling interval, each sample will contain fewer data samples, while many sample sets will contain no information sample data. This may also introduce higher computational complexity for calculating more sample sets.

Figure 1.3. The time series signal with a fixed TS sample size and converted into frequency domain by Short-Time Fourier Transform (STFT). Adapted from [8].

Therefore, a multiple resolution level in time and frequency domains Wavelet Transfer (WT) can be applied for analyzing the signal response in different resolutions. In the first level of WT, the analysis has high resolution in the time-domain but with coarse frequency resolution, while the analysis in the fourth level has fine frequency resolution with coarse time resolution. The blue cell filled in the contour plots of WT in level (1,4) has different resolution in both the time and frequency domains in every level. Figure 1.4 demonstrates the WT analyzing the signal in different resolutions in both time and frequency domains by the different levels of transformation.

Figure 1.5 demonstrates that the WT in level 1 to 4 of a signal contains two frequencies: f1=32 Hz from time= (0,2) and f2= 125 Hz from time= (2,4). The top plot is the WT in

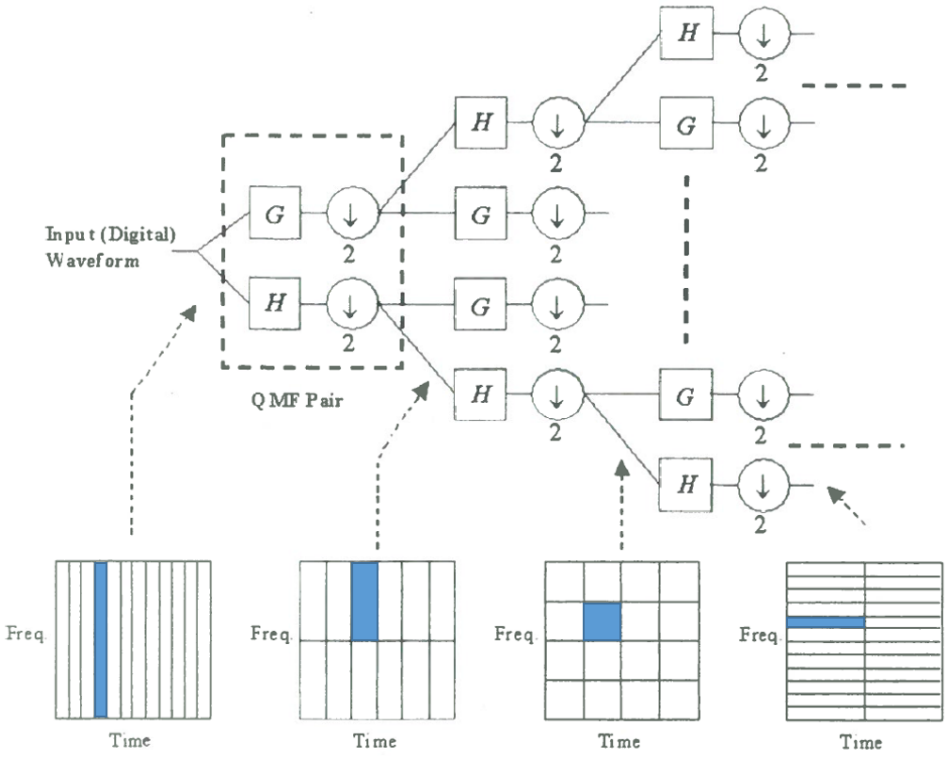


Figure 1.4. Four levels decomposition tree of wavelet transform. Adapted from: [9].

level =1. It has good time resolution that indicates the signal transition time = 2 (s), while the frequencies of the two signals cannot be clearly resolved. As the level of WT increases, such as the bottom plot in Figure 1.5, the WT in level 4, two signal frequencies are able to be resolved but the time domain resolution is not as high as the level 1 analysis.

The concept of multi-resolution level compensates for the drawback of fixed resolution STFT that does not have enough precision on the analyzed results if the improper TS is chosen. Although in this frequency-hopping signal detection study the frequency transition state is unknown and it requires a comprehensive time and frequency resolution level to extract the time of state transition, the varying TS concept presents a solution for higher-precision results with lower computational complexity. The goal is to have analyzing time matching the time of state transition yielding high precision results and preventing redundant computation in no-event transition periods.

Computer-based modeling for combat simulation has become an important role in the

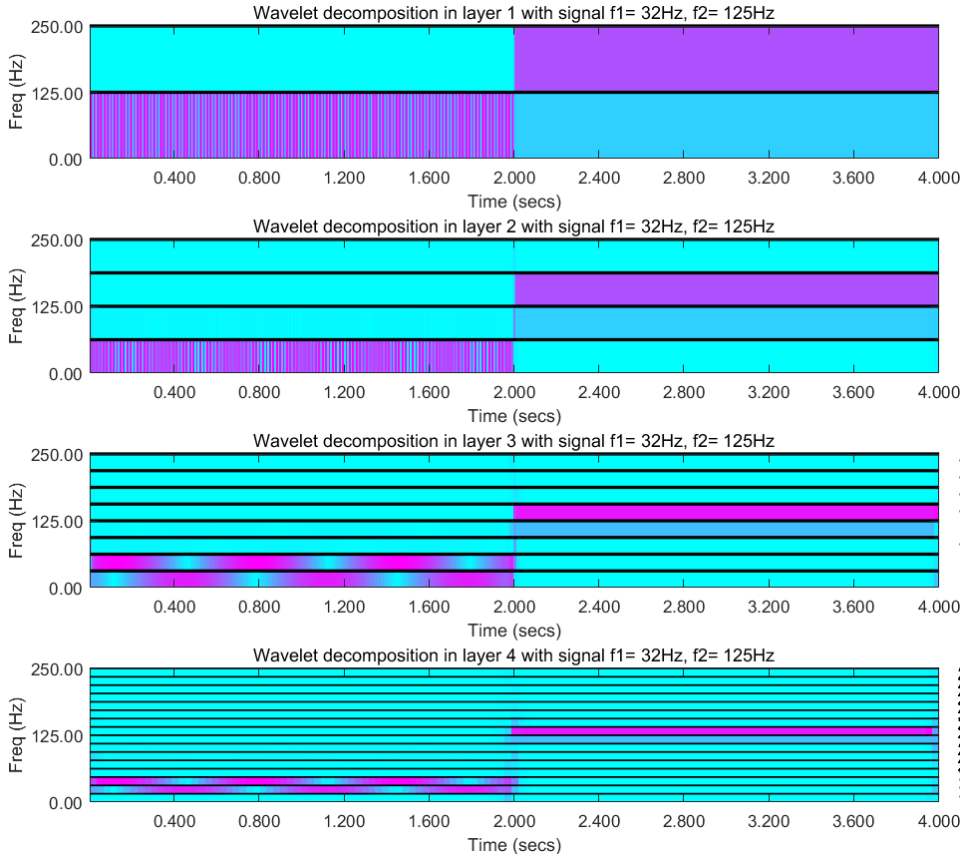
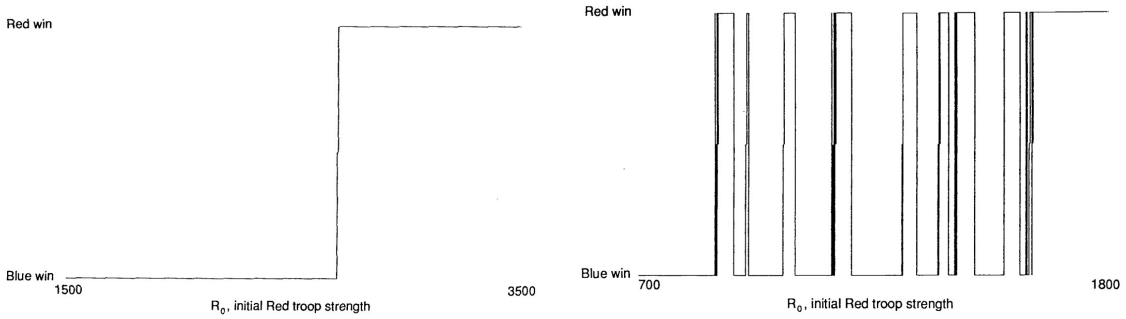


Figure 1.5. Four levels decomposition of wavelet transfer of a signal contains two frequencies f1= 32 Hz at Time=(0,2) and f2= 125 Hz at Time =(2,4). Adapted from: [10].

analysis of military training, strategy, tactics, and policy. These models can be most rigorously tested and validated to achieve high validity of the attempted simulation. Due to the model size and the complexity, it is almost impossible to fully understand the underlying dynamics of the models, and such models may subsequently yield the occasional disturbing nonintuitive result. Non-monotonicity and chaotic behavior of a deterministic combat model based on TS granularity has been studied by Dewar et al. [11]. For example, a Lanchester salvo model can simulate the attrition and the end of battle for two forces, blue troops and red troops. Under certain TS granularity, the model shows a non-monotonicity result as the monotonic initial troop strength is applied on the red-troops force. Figure 1.6 shows the end of battle results between two forces with monotonic initial troop strength on red troops in different TS granularity. In (a), the result has monotonic probability of win

for red troops as the initial troop strength is rising. However, the result does not match the expectation of the strength increment of red troop in (b). In some scenarios, the increment of the initial strength of red troop leads to the failed outcome for red troops as the win outcome happens with less initial strength of red troops.



(a) Monotonic result of red force to win. (b) Meta stable, non-monotonic result of red force to win.

Figure 1.6. Exemplar results of a Lanchester salvo model between red force and blue force. (a) shows the monotonic result as the initial strength of red force has monotonic increment. (b) demonstrates the non-monotonic result as the initial force strength of red force is increasing. Source: [11].

There are some possible causes of the non-monotonicity that deserve consideration:

1. **Model misconstruction:** The model is not constructed correctly in its original design attempt. In other words, these are coding errors when building the model. This is the most straightforward issue, and the non-monotonicity can be solved by correcting the coding errors. This is the "verification" of constructing the simulation, the first "V" of VV&A.
2. **Round-off and precision:** Computer calculations are subject to errors because of how computations are rounded off by the machine and how many significant digits the machine carries in the computations. The errors may not seem significant in a single variable. However, in a complex model with many interaction behavior factors, the small errors can lead to chaotic behavior that leads to a chain reaction with other factors.
3. **Time-step granularity:** The discrete time steps over which the attrition is computed correspond to a discrete approximation of a continuous "average" attrition in this salvo model. The size of the chosen TS and the goodness of the approximation have serious implications for the outcome.

Through these undesirable factors, model chaos can be introduced into the simulation results even if the simulation conditions are setup correctly. The chaotic behavior of the model results in misleading outcomes of which the practitioners may not be aware.

As another cautionary example, Figure 1.7 shows the engagement footprints of a Surface-to-Air Missile (SAM) simulation with deterministic detection and engagement processes. This example shows the fixed detection ranges produce erroneous outcomes in a complex situation. In the scenario on the left, the aircraft has higher Radar Cross Section (RCS) that leads to a larger high Probability of Kill (PK) zone (red area) to the SAM. By luck, the engagement of the missile to the target happened twice in the low PK zone (green area) and the target survived. Nevertheless, in the right-side scenario, the aircraft with lower RCS resulting in smaller high PK zone is nevertheless intercepted by the SAM at the same waypoint. The result is because the target being tracked a little later in the second scenario than the high RCS one, and that the result the one expected for the low RCS target. However, such simulation results present a unexpected outcome for the simulation.

Lucas [2] shows examples of biases resulting from deterministic approximations of stochastic events. A deterministic model is one in which a given input will always produce the same output, while a stochastic model is one in which the results are determined by using one or more random variables to represent uncertainty about a process or in which a given input will produce an output according to some statistical distribution. There are some reasons for utilizing the deterministic model:

- A good point estimation is sufficient for some simulations. Point estimations provide "the most likely" or "average" outcomes. However, a point estimate from a deterministic model is probably biased and possibly mistaken.
- Deterministic models require less processing. Unlike stochastic, a model requiring multiple replications for obtaining precise estimates for a given input set, a deterministic model has fixed output for each setup.
- Deterministic models are usually simpler. Adding necessary parameters to a model costs in processing time and understanding, and good analysts try to keep models as simple as possible.

Nevertheless, the closed-form solution of a model often does not necessarily exist due to the complexity of the simulation scenario. Thus these combat model scenarios are inherently

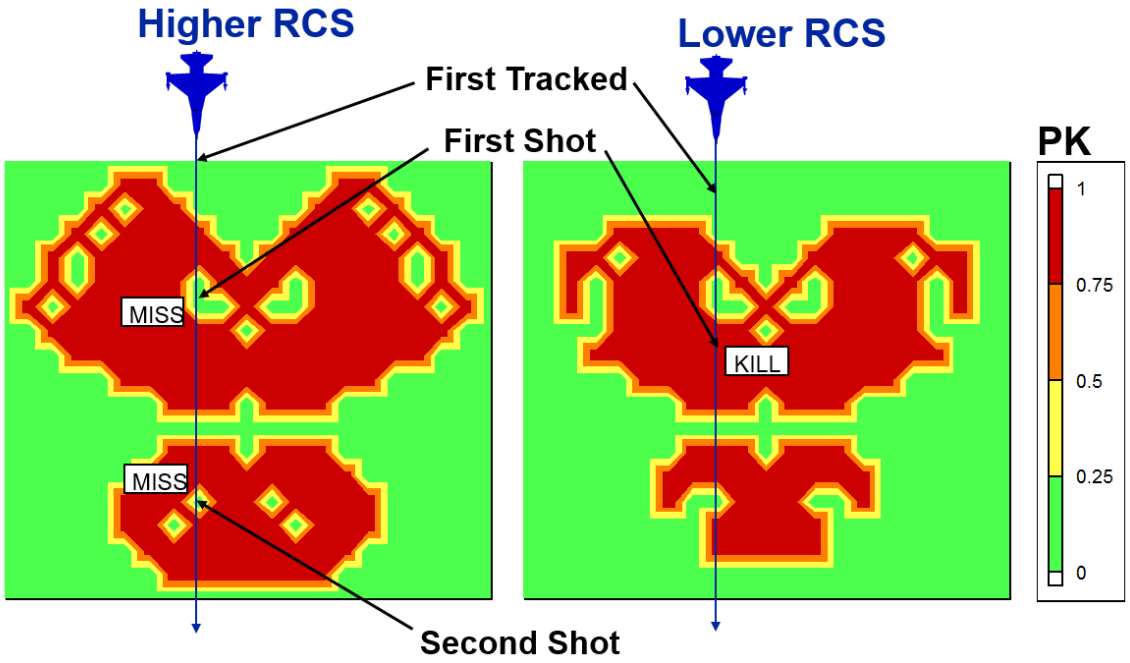


Figure 1.7. Two example scenarios of a SAM system simulation. In the left scenario, the target is fortunate to have the intercepts always occur in area of low PK even if the target has higher RCS and results in larger high PK area. However, the target in the right scenario has lower RCS leading to smaller high PK area and is intercepted. Source: [12].

stochastic. A vast number of factors have complex interactions that are difficult to determine precisely because of the processing uncertainty, future uncertainty, and decision uncertainty. These factors make it is impossible to model a detection event precisely. The attraction and detection events can be estimated statistically, and uncertainties are often quantified with Confidence Interval (CI). The benefits of applying stochastic models in combat simulation are:

- Stochastic elements tend to reduce instability. The non-monotonicity errors of the model can be suppressed by introducing parameters with Random Variable (RV), which can average out some of hard breaking points in models.
- Monte Carlo models explore more regions of feasible model space. Monte Carlo methods typically generate many more diverse trajectories through a simulation model space, while a deterministic model uses only the means of these RVs.
- Deterministic models underestimate the total variability by omitting the variability

of stochastic factors. Most combat models contain many random elements, which contribute to the overall variance in outcomes. The total outcome variability increases as the number of stochastic variables included in the model. Deterministic models do not account for this variability, instead applying only an average of these factors.

- Monte Carlo methodology facilitates dealing with uncertainty and complexity. Even a simple combat model makes the analytical solution prohibitive. If the uncertainties are represented by RVs, this structure allows random samples of potential outcomes to be extracted by the Monte Carlo approach.

For a radar-targets search simulation scenario estimating a radar's Time to Detect (TTD) against hostile targets, numerous factors can contribute to model performance such as target's distance, antenna sweeping period, aspect angle of targets, and target speed. Some parameters are sensitive to the performance of the whole system and a rigorous computation is often too complex to accomplish in a real-time simulation, such as RCS of targets from different aspect angles.

Wang et al., in their study of constructing radar system simulations, propose a *universal program framework* that considers the radar-target scenario using an object template. The template divides the simulation into three subsystems classified by functions. These functions include: 1) scene target echo-generation system; 2) radar information process system; 3) radar control, display, and estimation system [13].

A recurring problem arises in approaches to dynamic target-sensor scenarios, when a sensor's detection algorithm is based only on the range between sensor and target [14], or a sensor's detection perimeter [15]. Both scenarios neglect the dominant factor of aspect angle of targets, since the change of aspect angle can lead to drastic variations in RCS response. Similarly the maximum detectable range of sensor derived from the Radar Range Equation (RRE) in Equation 2.3 also fluctuates along with aspect angle. RCS can have a huge influence on received signal power at the receiver, and has been recommended as a topic of future work in radar M&S research by Inggs et al. [16].

Wang's study [17] has incorporated RCS values into modeling by a ground clusters numerical Weibull model in a radar M&S focusing on Terrain Environment effects. The cluster backscatter RCS is the function of terrain range and aspect angle relative to the sensor in the Weibull model that has been previously measured and built in prior research.

This simulation applies the cluster RCS effect with aspect angle consideration, which has a different RCS value as the object location varies in the simulation.

However, when integrating a non-parameterized RCS model generated from an arbitrary target, a simulation application might produce incoherent responses due to considerable computational latency from generating target RCS. In addition, TS size in a simulation plays a crucial role in contributing to the accuracy and efficiency of simulation results. For more detailed resolution purposes, the TS size has to be increased, while it also raises up the simulation computational complexity and introduces insignificant data into the system. Nevertheless, the precision of the result can only reach a certain level and can never be exactly accurate. It is clear that better methodology is needed.

1.3 Problem Statements

To sum up the advantages and disadvantages of various simulation approaches from the case studies, these are the characteristics of different types of models:

1. Physics models:
    - Advantages: Physics models provide high accuracy and high fidelity for representing events in the real world. When compared to statistical distributions or simplified (cartoon physics) models, differential-equation based physics models contribute to the fidelity of the simulation.
    - Disadvantages: Physic models are typically complex and cumbersome due to the inherent complexity of the real objects. The potential drawbacks of the physics model in the simulation are the corresponding computational complexity of model structure and the incoherent simulation timing between each component.
2. TS models:
    - Advantages: TS models are relatively intuitive to build. The concept for computing the next event based on a fixed time increment is familiar to most simulation practitioners. Furthermore, model algorithmic resources are abundant. The majority of simulation applications are built using TS structures.
    - Disadvantages: Simulation efficiency and precision come as a high cost due to the granularity of TS. Utility typically degrades either the efficiency or the precision in the hybrid when choosing different TS resolutions to work together.

3. Discrete-Event Simulation (DES) models:
    - Advantages: Simulation precision is high. Opposite to the step size granularity issue of TS models, the DES models are executing precisely at the state-transition event time. Model efficiency is also high. Since the simulation calculation happens only at the state-transition time, there are no irrelevant events to be considered, which reduces the redundant computation in the simulation.
    - Disadvantages: Low model fidelity of physical phenomena is commonplace in current combat-modeling simulation applications. Because the DES provides much less model fidelity for physical architectures such as sensing then in the TS models, the available sensor models for combat modeling seriously limit overall model fidelity. They also raise major questions regarding simulation validity.

1.3.1 Barriers and Gaps

For constructing a high-fidelity simulation with complex behaviors, such as a dynamic position sensors-targets simulation, there are barriers for model construction.

1. **Model execution efficiency:** The model efficiency degrades due to the divergent computational latency factors integrated into the simulation.
2. **Time-consuming calculation:** To generate the highly detailed physics models, comprehensive calculation is needed for a physics models with high resolution.
3. **Model architectural complexity:** The model complexity rises along with the increment of the number of factors are considered into the systems.

Potential solutions for filling these gaps are:

1. **Offline computation:** For high computational-complexity models, if possible offline computation is needed for isolating the tremendous computational latency for other factors.
2. **Applying high-performance computing (HPC) resources:** To shorten the time to compute highly complex model calculations, High Performance Computing (HPC) such as cloud servers provides a possible solution if the model can be constructed in a parallel form.
3. **A new flexible modeling architecture:** As efficient and flexible modeling architecture is required for approaching higher resolution and more complex

simulation scenarios.

1.4 Contributions

In a target-acquisiton search-radar simulation scenario, the primary information obtained from the sensor entity is the time and the position of a detected target. Therefore, the TTD of the target to the sensor, which is applied to resolve the detection location of the target, is the key information needed for the sensor entity in the simulation. Buss et al. [15] propose a DES constant-rate sensor, which determines only the TTD of the target when successful contact between the target and the sensor occurs. The sensor model detection behavior is efficient based on only two events: 1) whether the detection happens and 2) when TTD occurs. However, the detection algorithm of this DES is too crude to consider the target position and aspect angle variation in a dynamic sensors-targets simulation scenario. Methodology improvements are possible. The contributions of this dissertation follow.

1.4.1 Accomplish an Efficient DES Sensor Model with the Consideration of the Range and the Aspect Angle of the Targets based on High-Fidelity Physics Models

The primary contribution of this study is to construct a high-fidelity sensor model in concert with a highly efficient DES architecture.

Figure 1.8 depicts the goal of this study, which is to construct an efficiently executing DES architecture meta sensor model with the same detection reports as the constant rate sensor: 1) whether the detection happens as determined by Probability of Detection (P_d), and 2) when the TTD occurs, with the simulation incorporating high-fidelity physics models into the sensor-detection algorithm. A two-stage approach is used to achieve this highly detailed DES architecture sensor model:

1. **Constructing a high-fidelity sensor model using a hybrid model architecture:**
   This model incorporates highly detailed physics models into the sensor detection algorithm.
2. **Forming a meta-model with statistically identical performance to the hybrid model:**
   An associated meta-model learns the hybrid model detection algorithm from the

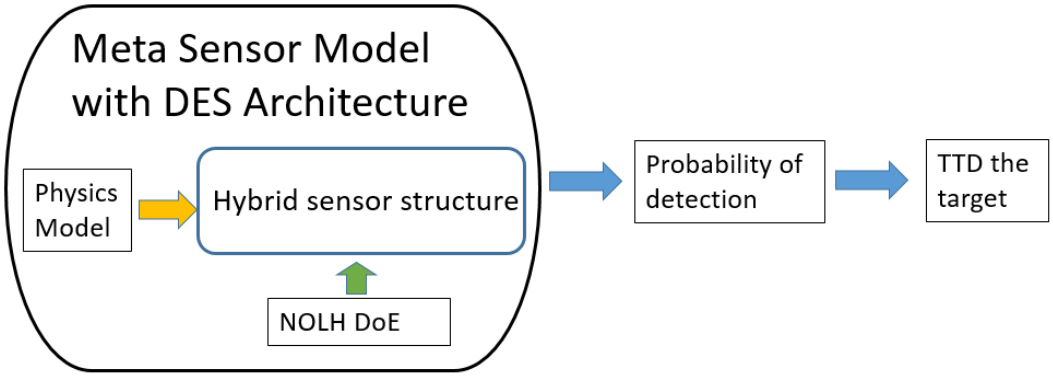


Figure 1.8. High-efficiency and high-fidelity radar system simulation flowchart.

observation data generated by the efficient Nearly Orthogonal Latin Hypercube (NOLH) Design of Experiment (DoE) results.

1.4.2 Developing a Hybrid Sensor Architecture, which Incorporates High-Fidelity Physics Models with Different Simulation Timing Mechanisms: Atemporal, DES, and TS, into a Coherent Simulation Timing Model

To integrate highly detailed physics models with divergent run times into a simulation, a hybrid sensor structure that incorporates models from three categories with coherent simulation timing is proposed in the study. The experimental flow chart is shown in Figure 1.9. The three categories of the hybrid sensor model are: 1) atemporal construction, 2) discrete event simulation construction, and 3) TS construction.

The hybrid sensor model integrates the high-fidelity physics parameters into the RRE, which considers the dynamic position between sensors and targets through simulation with a comprehensive detection algorithm. Because of the variety of characteristics in radar parameters, each of them has preferable and efficient methods to apply during the simulation. In this study, all the considered radar equations are classified by the interaction ability between entities. Those parameters having no interaction with other entities, which can only be calibrated by radar design attempts, are classified as the controllable parameters and are implemented in a specific initial value in the DoE. However, the parameters in the RRE with the interaction relation between entities have to be computed as simulation state

changes, such as RCS. Those parameters cannot be designated as the initial value in the simulation, but they will be updated as the simulation is proceeding. Unfortunately, due to the different characteristics between these two categories of the parameters, the modeling structures associated with each type of them are different. The simulation time mechanisms are also incoherent in different categories. This work constructs a hybrid model with three modeling constructs that mitigate the disparate simulation time mechanisms of each paradigm.

1. **Atemporal model:** The atemporal models have no time factor of the model response. With the highly detailed physics characteristic, however, these cannot be interaction coherently in the simulation with other factors due to the tremendous computational latency, such as the target's RCS value. The RCS value varies along with the aspect angle between the targets and the sensor in dynamic simulation scenarios. The RCS is highly sensitive to the geometry of the object, and the projection of the object changes as the aspect angle varies. Due to the high computational latency, these factors are classified as atemporal models and need to be processed separately from other modeling structures. Interestingly, since the original phenomena (such as electromagnetic response) required time-aware offline simulation, the resulting response model can be thought of as having temporal response compiled or distilled out of the results.
2. **DES model:** DES allows for precise and efficient event execution. As previously discussed, the major limitations and drawback of TS simulation are the simulation precision and execution efficiency. A simple detection algorithm DES sensor model structure is applied as a framework for new sensor models to integrate more radar detection-related parameters. This is intended to increase the detail of the radar model detection algorithm while the simulation can still maintain the precision level in the results with relatively higher execution efficiency than the pure TS simulation structures.
3. **TS simulation:** Differential time step model structure is used to represent inherent TS behavior. This model structure emulates radar antenna mechanical sweeping behavior. The attempts of target detecting by a search radar happens once for every antenna scan at the target in a period of scanning rate. The antenna scan period is proportional to the target integration gain in the RRE but inversely proportional to

the rate of updating the target position during simulations. The longer the antenna sweeping integer, the higher the target integration gain, and the lower the target updating rate.

1.4.3 Generating the High-Fidelity and Highly Efficiency DES Sensor Model by a Meta-Model, which is Statistically Equivalent to the Hybrid Model with Much Lower Modeling Complexity

Despite the hybrid sensor model integrating the incoherent modeling factors into one modeling structure efficiently and coherently, the overall model structure is still bulky for carrying all parameter information. The simulation execution efficiency is not optimal either, due to the embedded TS structure. The execution efficiency is affected heavily by the size of the TS. In most cases, the scanning events are processing no state change scenarios repeatedly and these are the inherent behavior of TS structure. For forming a more computationally efficient sensor model with the equally likely physics-based detecting algorithm sensor, the second-stage process is applied for creating a simplified structure sensor model. Forming a meta-model and generating results is relatively simple compared with the baseline hybrid sensor model, which is the given DoE [18]. The highly efficient DoE for analyzing the detailed interaction information between each parameter that has been considered in the hybrid sensor model is generated by NOLH design. The DoE allows the hybrid sensor model behavior to be studied comprehensively by the efficiently space-filling Design Point (DP)s to cover the range of parameters of interest in the sensor model with a concise number of DPs for a reasonable simulation execution loading. The meta-models are fit by the regression analysis with the observation data from the simulations conducted by the DoEs. The meta-models undergo multiple verification processes to prove the validation of the meta-model to the baseline hybrid sensor model.

The properly fitted meta-model encapsulates parameters that have been considered in DoEs without carrying bulky computational loading for simulation applications. Nevertheless, it remains important to note that the meta-model is a smoothed representation of the simulation experimental outputs. This derivation has a tendency to hide unusual occurrence (i.e., outlier events) that can occur in stochastic simulation, and indeed in the real world as well. With three categories of modeling structures and the two-stage model construction processes, the radar sensor models can not only integrate the model with high physical detail

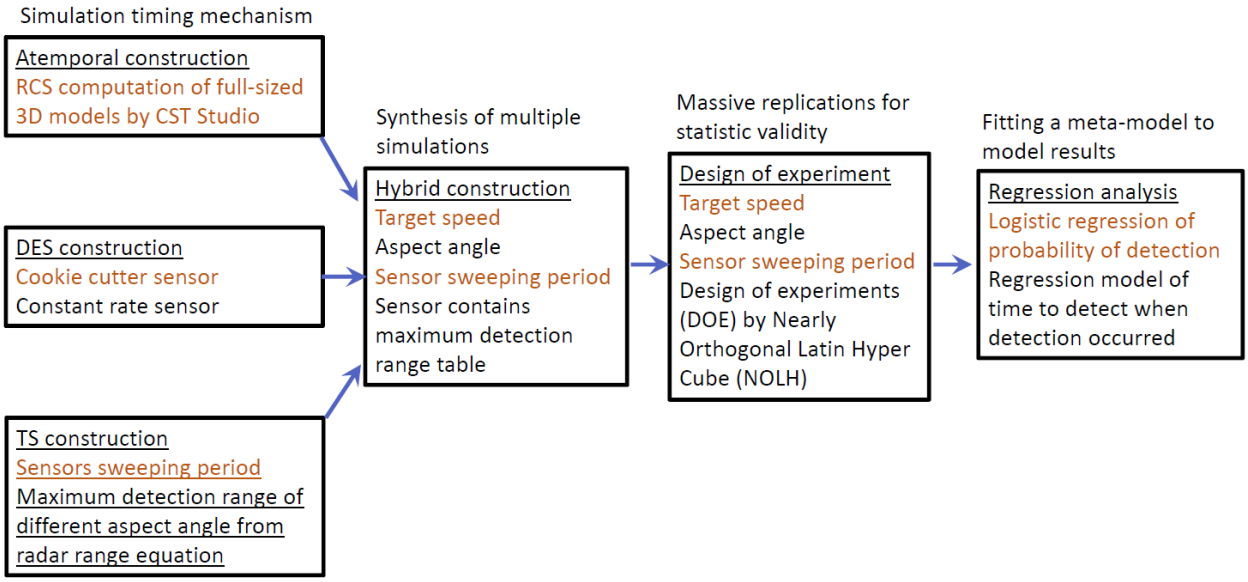


Figure 1.9. Methodology roadmaps: experimental-design flowchart showing hybrid sensor models integrated from three simulation categories, followed by overall regression analysis.

radar parameters into detection algorithms but also generate a relatively low construction and computational complexity meta-model with validating behavior to the baseline hybrid model. The meta-model predicts P_d and TTD. The results make the sensor model working as the most succinct DES structure with considering the target range and aspect angle, which is not considered by the DES sensor model before.

1.5 Dissertation Organization

This dissertation is organized as follows: Several important related works which support and inspire this research are discussed in Chapter 2. These works provide a general knowledge of DES and the construction of rudimentary and efficiently executing DES sensor models. In addition, the fundamental radar principle and the parameters that have been considered into the detection algorithm of new hybrid sensor model are also introduced. In this section, the significant effect of radar maximum detection range by the variation of RCS in different geometry of object is analyzed.

In Chapter 3, there three prototype advanced sensor model architectures are introduced. Two-tier and multiple-tier probability of detection DES sensor models are proposed. These

two models illustrate progressive thought on migrating from a basic DES sensor model to a more complex model. The basic DoE and data analysis process are applied for the rudimentary trial approach of further sophisticated sensor models construction and advanced DoE design and simulation analysis.

The advanced hybrid framework sensor model is introduced as the third model. To ensure that maximum detail is incorporated into model parameters, a hybrid model that integrates atemporal, TS structure into DES is used. This demonstrates the solution hybrid sensor construction to incorporate the incoherent simulation time mechanism of models from three categories. The event diagrams of the hybrid sensor and the simulation event schedule tables are used for illustrating the structure of the hybrid sensor model.

Chapter 4 illustrates the principle of DoE for conducting the simulations. For the regression analysis, which is the second-stage of the model efficiency optimization process, comprehensive and efficient DoEs for analyzing the hybrid sensor model behavior are necessary. The DoEs applied in this research are set up efficiently using NOLH from Cioppa [18] and are introduced in this section.

In Chapter 5, the fundamental principles of linear regression analysis and logistic regression analysis are introduced. The linear regression analysis is applied for predicting a generalized TTD of the hybrid sensor model based on the observation data generated by the DoEs, while the logistic regression analysis is for predicting the P_d of the sensor to detect the targets. For the validation analysis of the fit meta-model to the baseline hybrid sensor model, a series of model selection, model shaping and model verification methodologies are introduced. Cross Validation (CV) is applied for making sure the fit meta-models can properly represent the hybrid sensor models in particular. As the result, new numerical meta-models are generated that incorporate the parameters considered in the hybrid simulation framework.

A summary, conclusions, and recommendations for future research are in Chapter 6.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Related Work

Human beings, who are almost unique in having the ability to learn from the experience of others, are also remarkable for their apparent disinclination to do so.

–Douglas Adams [19]

2.1 Physics-Based Web 3D Model Generation and Animation

Cheng et al. [20] produced a study that applies Matlab and Simulink creation and animation of Extensible Three-Dimensional Graphics Standard (X3D) in web-based simulation. Matlab is a powerful tool to compute high-fidelity engineering models and graphically plot the results. Simulink implements Matlab .m source code into block diagrams and flow charts to execute the simulation. The project demonstrates how physics equations implemented in Simulink can animate X3D or Virtual Reality Modeling Language Standard (VRML) models, along with the methods to convert Matlab .fig format into an X3D object so we can apply it into web-based animations. The overall operation flow chart is shown in Figure.2.1 and the processes are listed as follows:

1. The "Matlab.m" block implements the mathematical physics model in Matlab.
2. In the "Matlab.fig" block, the 3D model figure is plotted as *.fig based on the mathematical model implemented by Matlab.
3. Using a built-in figure-conversion library in Matlab, the 3D *.fig models are converted into two widely used web-based 3D object formats: "VRML.wrl" and "X3D" since these two formats are directly similar and capable of being converted from one to the other.
4. "Simulink Simulation" animates the "VRML.wrl" objects by physics equations and simulations can be shown in "VRML Players" or "HTML" web browsers.
5. The "X3D" models are also to be displayed in "HTML" web browsers.

2.1.1 High-Fidelity 3D Model Generation

In the first part of this demonstration, a phased-array antenna radiation pattern is calculated from Matlab based on the antenna theory and then converted into web-based 3D models,

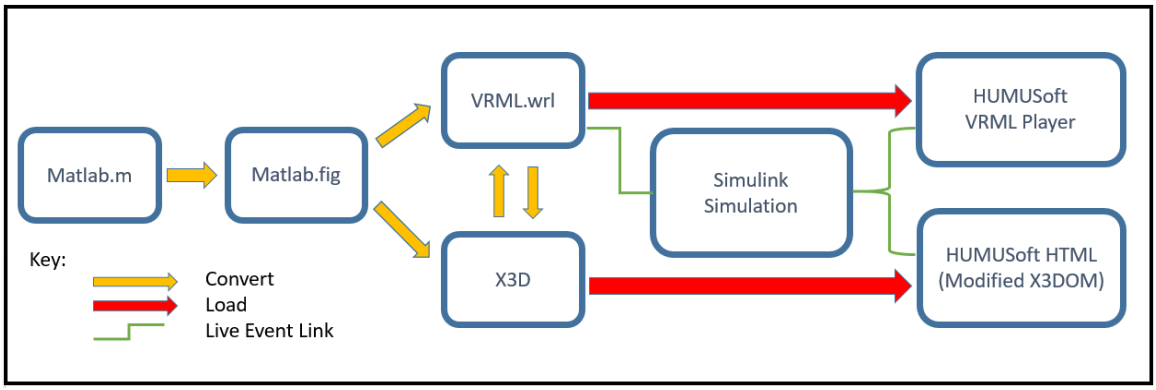
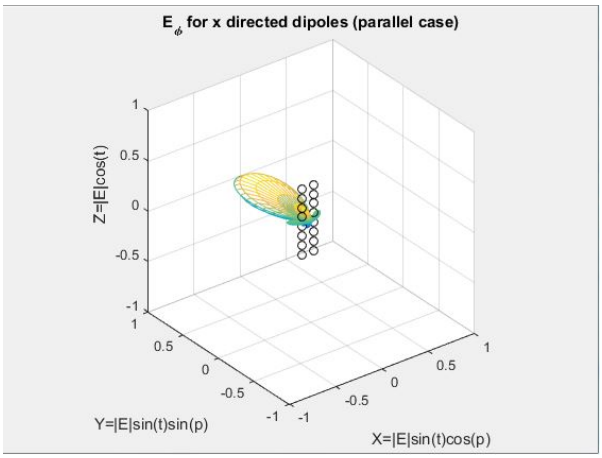


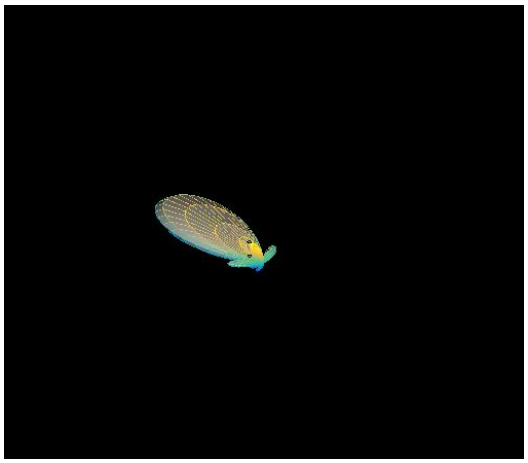
Figure 2.1. Design patterns for Matlab and Simulink creation and animation of X3D in web-based simulation.

such as .wrl, .x3d scenes for further application. The phased-array antennas are the antenna set assembled by several antennas. These antennas are arranged in space and interconnected to produce a directional radiation pattern. The array elements interact with each other and alter the current from that which would exist if the elements were isolated. The interaction, called mutual coupling, changes the current magnitude, phase, and distribution on each element. This is manifested by a total array pattern, and these effects depend on frequency and scan direction [21]. A 3D beam pattern of a seven elements phased-array antenna is generated by Mathworks Matlab based on linear array antenna electric field computation and is shown in Figure 2.2a. For wider application on web-based simulation, the high-detailed Matlab.m 3D figure is converted into a web-based 3D model .x3d format shown in Figure 2.2b, .wrl format in Figure 2.2c, and .HTML format in Figure 2.2d. X3D is the open international standard for 3D on the web. X3D is used for building 3D models from simple to sophisticated constructions. X3D can show animated objects from different perspectives, and allow user insight and interaction. The X3D models can be combined and connected to build sophisticated 3D virtual environments running in a web browser, locally or across the Internet [22].

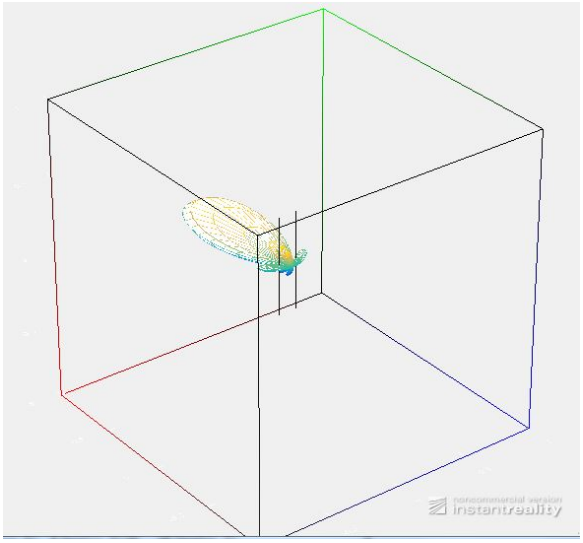
This study demonstrates an approach where a highly detailed and physically based model generated by Matlab can be extended to other popular web-based 3D objects for Networked Virtual Environment (NVE) application purposes, such as X3D and VRML objects. With the benefit of computing and generating the objects with an efficient engineering application tool Matlab, which has a variety of technical libraries support, the web-based 3D models



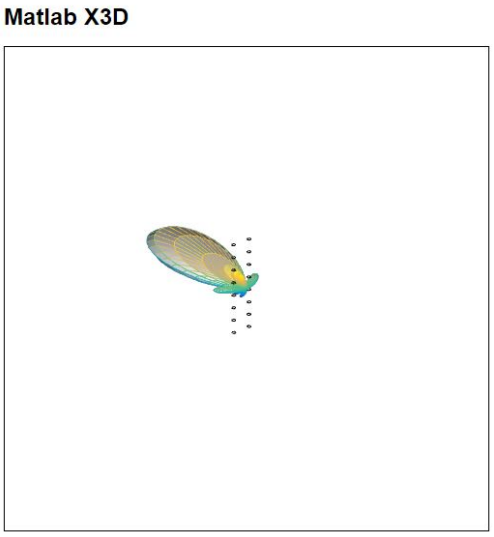
(a) Antenna Beam Pattern Model in Matlab.m



(b) Antenna Beam Pattern Model in .x3d



(c) Antenna Beam Pattern Model in .wrl



(d) Antenna Beam Pattern Model in .HTML

Figure 2.2. Phased-array antenna beam pattern model created in Matlab .m source and converted to .x3d, .wrl, and .HTML models. (a) Source: [23].

in NVE are no longer just for "pretty" visualized object demonstration, but also display the results based on the accurate assumptions in the simulation scenarios. Detailed model generation and conversion procedures are provided in Appendix A.1.

2.1.2 Physics-Based 3D Model Animations

The second stage of the physics-based visualization study conducts a dynamic free-fall object scenario. The simulation is animated by Simulink, which is the real-time simulation toolkit of Mathworks Matlab [24], in a fixed TS $\triangle t$. The location of the object to present the moving behaviors is based on the free-fall dynamic equations with G-force = 9.8 (m/s). The object velocity $v = G \cdot \triangle t$ [25] and the velocity change of the object during the simulation is shown in the bottom of Figure 2.3. The moving distance within time $\triangle t$ is $\triangle d = \frac{1}{2} G \cdot \triangle t^2$ [25] and the object position is monitored and shown at the bottom right of Figure 2.3. The resilience force F of the X3D object bouncing behavior is implemented by Hooke's Law. $F = -k \cdot \triangle s$ [26] while $k = 600$ is a constant factor characteristic of the spring and $\triangle s$ is the compression distance of the cube. The resilience force is proportional to the "squishiness" of the object during the contact of the ground and it is shown at bottom right of Figure 2.3.

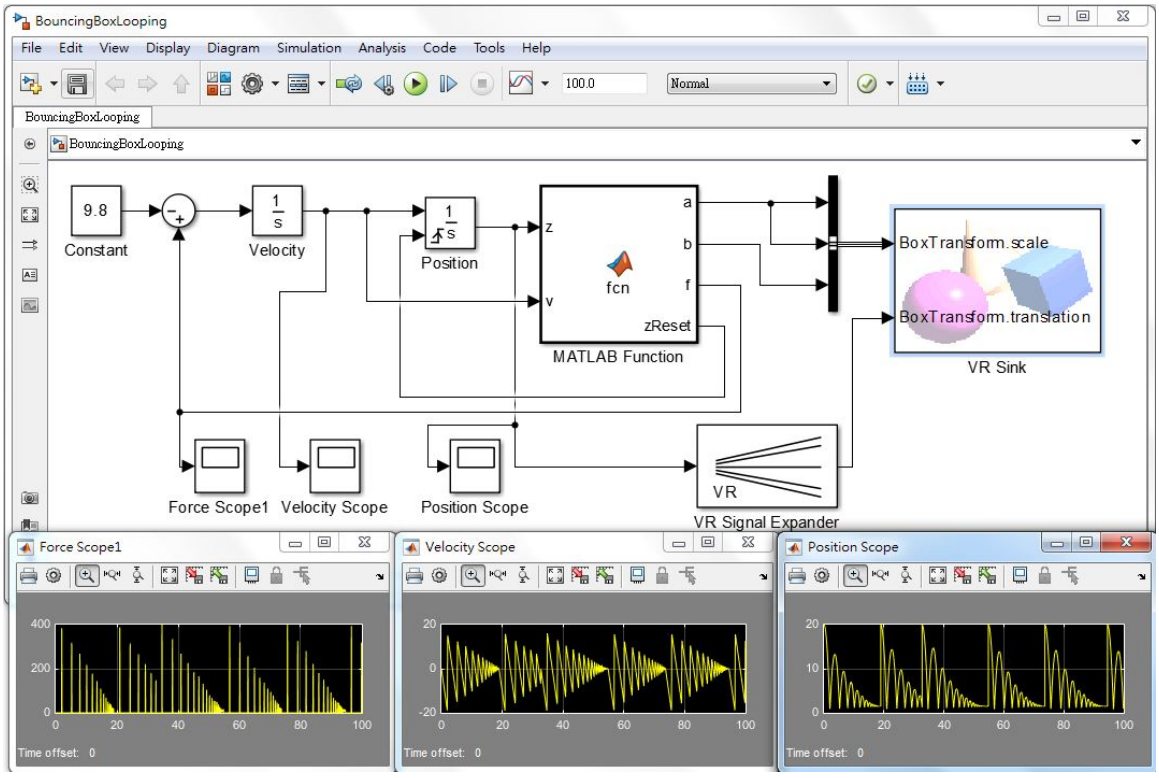


Figure 2.3. Block diagrams of Simulink implementation with plotted position, velocity, and force variation of a bouncing/springing box.

The Attenuation Force (AF) is applied for a more detailed simulation scenario to represent the fractional loss during compression and resilient process in the real world bouncing object behavior. $AF = -6 \cdot v$, while the -6 is the attenuation factor. Therefore the attenuation can be observed through the object movement monitors in Figure 2.3.

The shape of the red X3D box object is slightly expanded along x, y axis while the squishiness is compressed along the z axis. In Figure 2.4, the object shape is transformed from a squared cube into a flat box (center plot) and then resumes its original shape after bouncing (right plot) with the scaling parameter $a = \sqrt{\frac{1.5^3}{\triangle z}}$ along x, y axis. Linear deformation effects are assumed. Due to the accurate physics assumption for the animation response, the scenario looks reasonable and close to the impression of real world physics movement. The block diagram of the Simulink animation is shown Figure 2.3 and the detail simulation construction is described in Appendix A.2.

This study has demonstrated an approach of 3D model construction based on highly physics-detailed equations, which are computed by the efficient engineering simulation tool and converted into popular NVE formats for wider simulation application. Also the animation is rendered based on the physics movement equation by a well-developed real time engineering simulation tool. However, the simulation precision and computation complexity are still highly associated with the size of TS during the simulation. The finer the TS, the higher the details of the simulation results and the higher the computational complexity of the simulation.

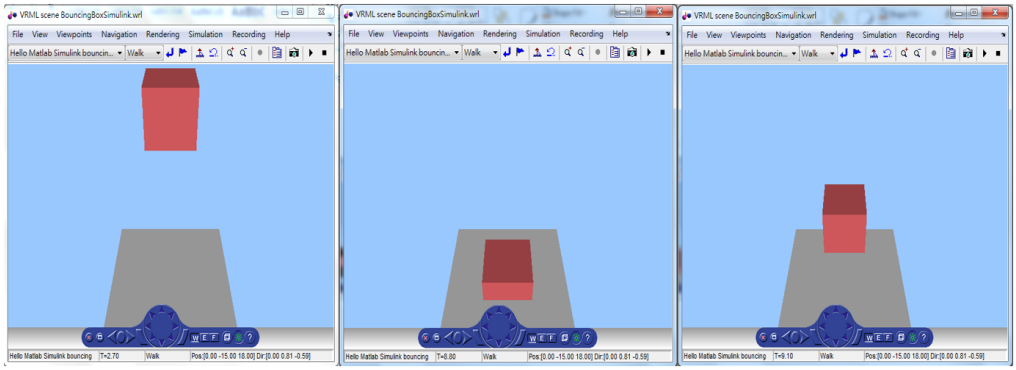


Figure 2.4. The X3D bouncing-box animation simulation rendering by Simulink. The screen shots of object movement display three states: free fall (left), compressed (center), and post-bounce (right).

2.2 Discrete Event Simulation (DES) Structure

A simulation model is a representation of an actual system and the activities that happen in the system. A simulation advances time according to the model architecture. A TS simulation, advances time according to a fixed time increment. The TS may be too coarse to result the accurate information or too fine to introduce redundant computation into the simulation. A DES model, however, advances simulation time from event to event. The time between events is typically variable and the events can occur at any time. The DES structure is illustrated in this section.

2.2.1 Concepts and Components of DES Models

The basic concepts and components regarding the construction of DES models are listed as follows [27]:

- System variables: There are two types of variables in a DES model.
  - Parameters: Parameters are variables that do not change during a simulation.
  - State variables: These variables are the collection of all information needed to define what is happening in the system. The determination of state variables is a function of the purpose of the system models. The state variable can be changed at any time during the simulation. The value of each state variable is a function of its initial value together with the sequence of state transitions that have occurred.

- Entities: An entity represents an object that requires explicit definition. An entity can be dynamic and move through the system, or it can be static and have interaction with other entities. An entity may have instance variables that pertain to the entity alone, which characterize the entity in the model.
- Event: An event indicates an instantaneous state transition during the simulation. Events are scheduled in the future according to model's design. Pending events are contained in a future events list and proceed one at a time, with time advancing to that of the next scheduled event.

2.2.2 DES Event Graph

The description of scheduling event in the basic DES framework indicates a directed, binary relationship between the event that is being processed and an event that is scheduled. Event graphs were introduced by Schruben [28] and are used to represent the activities in DES. An event graph consists of nodes and directed edges. A node corresponds to an instantaneous state transition. Each edge can have an associated Boolean condition and a time delay. Figure 2.5 shows the fundamental construct for event graphs and is described as follows: the occurrence of event A causes event B to be scheduled after a time delay of t by a Boolean condition (i), i.e., the event B is scheduled only if the condition (i) is true. Therefore, the basic event graph paradigm contains only two elements: the event node and the scheduling edge with two options on the edges (time delay and edge condition).

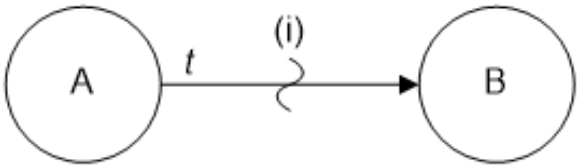


Figure 2.5. Fundamental event graph construct. Source: [29].

A Simple Example of the Arrival Process

Figure 2.6 is an example of an arrival process modeled by two events: Run and Arrival. There are two kinds of parameters in this arrival process model:

- **Parameters**
  - $\{t_A\}$: a sequence of (possibly random) times is between the occurrences of two events.

- **State variables**
  - N: the number of times the event has occurred at each state transition. Its initial value is 0.

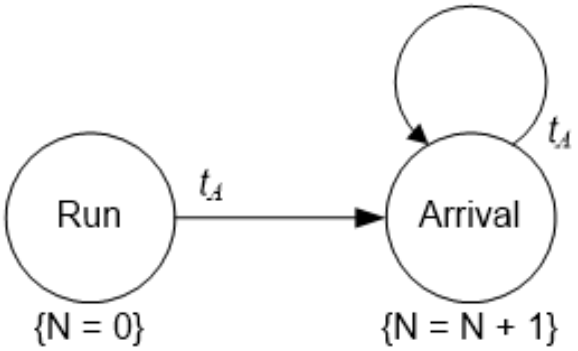


Figure 2.6. ArrivalProcess event graph. Source: [29].

The description of the arrival process is:

1. When the simulation starts at "Run" event, the number of the event counter N is set as 0 and the "Arrival" event is scheduled with the time delay t_A. If the parameter given by the sequence of interarrival times $\{t_A\}$ is random, each value of the time delay will be a different value.
2. The "Arrival" event has a self-scheduling loop with a time delay t_A to recursively schedule "Arrival" event.
3. "Arrival" event adds the increment of the event counter N by 1 at each time when the "Arrival" event happens. Therefore, N has an increment of 1 in every time delay t_A.

Canceling Edges

For the reason that the previously scheduled event is no longer valid, the canceling edge represents the removal of the scheduled event. Figure 2.7 shows the canceling edge to cancel the event "B", which is scheduled by event "A" under the condition (i). The canceling edge is represented by a dotted-arrow to indicate the direction of the event source.

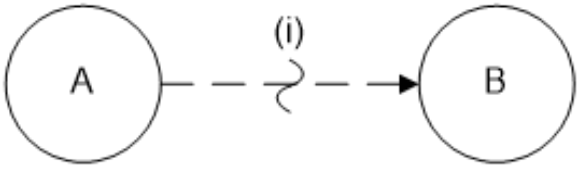


Figure 2.7. Event canceling edge. Source: [29].

Parameters on Events and on Arguments Edges

For situations in which the data or the arguments are required to be passed through the event transition to the next event, the passing arguments are denoted as j in a box on the scheduling edge in Figure 2.8. It is often useful for data to be passed to an event and it is much like passing an argument to a function or a method in computer programming.

A parameter on an event is indicated by a list of variables in parentheses. It is similar to the syntax of parameters on a method in computer programming. The argument on the edge is a list of expressions that match the event's parameters syntactically.

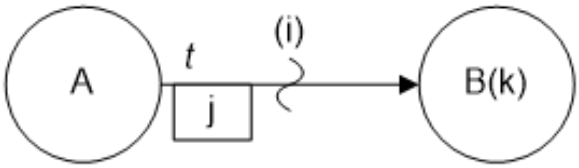


Figure 2.8. Scheduling edge with arguments and event with parameters. Source: [29].

2.2.3 DES Event Graph Components

When constructing a large scale model for which the entire model consists of a single event graph, the number of entities and scheduling edges make the event graph difficult to understand and maintain.

One solution is to define event graph components rather than to keep all functions in one graph. An event graph component is only responsible for its own state variable and it is not responsible for any other component's state variables. An event graph component has its own parameters, state variables, and operation algorithm to determine the outcome and timing of the state transition in simulation.

SimEventListener Pattern

In Simkit implementation, a listener pattern is called "SimEventListe ner Pattern" [30]. A simulation component that is interested in responding to simulation events that occur in other components is registered as a "SimEventListener" to those source components. The SimEventListener pattern is the mechanism to indicate which event in a component can affect the state of another. Figure 2.9 shows the representation of the SimEventListener pattern. The "Listener" component listens to all events, which happen in the "Source" component with a "fork-shape" mark to indicate the source component. If "Listener" has an event that is identical (in both name and signature) to the "Source" component and is scheduled, then the event will be processed in the Listener component as if the Listener component had scheduled it. By convention, "Run" events are not dispatched to listeners.

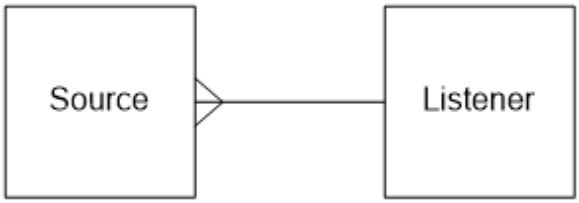


Figure 2.9. SimEventListener relationship. The "Listener" component "hears" all component "Source" events . Source: [29].

A SimEventListener Pattern Example

Figure 2.10 shows an example of the listening "Entity A" listens to the events from the source "Entity B" with the connection of a SimEventListener pattern. The event listener listens to the identical event with the same name and signature, such as "EndMove" in this example. The simulation starts from the "Run" event in Entity A and then schedules the "StartMove" event. When "StartMove" occurs, it schedules an "EndMove" event. Entity B has no "StartMove" event, so it does nothing until the event listener hears the "EndMove" from Entity A, and then the "MoveToWP" event is scheduled in Entity B. The arrow of the event listener indicates the direction of the source entity and the responding entity.

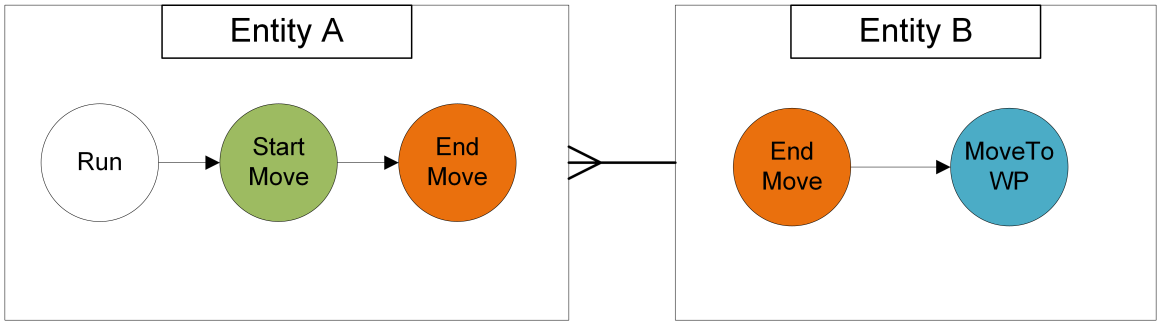


Figure 2.10. A DES event listener pattern example between two entities. Entity B listens for "EndMove" event from Entity A.

2.2.4 Precision and Efficiency of DES

With this event-driven characteristic, the DES construction has an advantage in computation precision and efficiency. Alrowaie [31] compares the accuracy of discrete event and discrete time in an example of the Euler method approach for solving the differential equation. In one dimension, a simple linear differential equation has the following form:

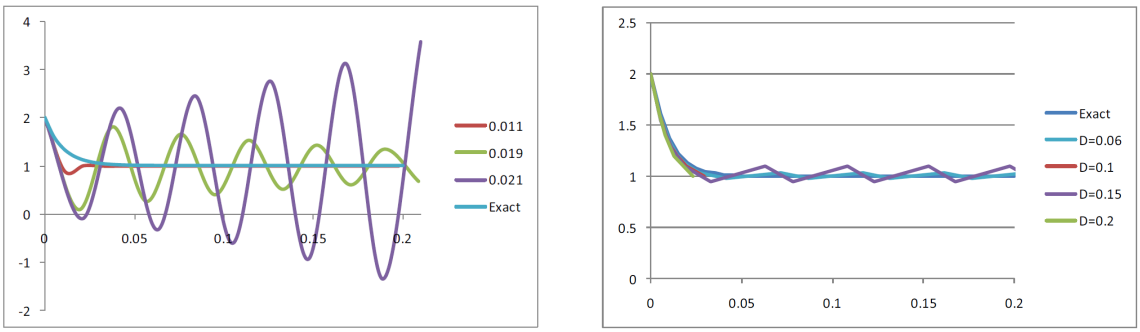
$$\frac{dy}{dt} = f(y,t) \tag{2.1}$$

The Euler method turns the differential equation into a difference equation by discretizing time into intervals of size $\triangle t$ and the Equation 2.1 is turned as follows:

$$y(t+\triangle t) = y(t) + \triangle t \cdot f(y(t),t) \tag{2.2}$$

Equation 2.2 is a first-order Taylor series approximation to Equation 2.1. The new state value is the old state value plus an increment, which is the old state value multiplied by a fixed differential time-step $\triangle t$. The resulting solution has significant variation for different size of $\triangle t$ and the difference is shown in Figure 2.11a. For the less-erroneous result, the smaller time-step size is required. However, from the perspective of computational complexity, the number of computation events has an exponential increment as the TS size decreasing. When TS size is decreased by half, for example, this introduces twice as many interaction states in each entity. The computation of the number of N states interaction increases to N^2.

An efficient DES approach to solve the differential equation "quantization method" was



(a) Euler solution errors in different size of $\triangle t$ (b) Quantized solution errors in different size of D

Figure 2.11. The estimation errors comparison: (a) Euler method, (b) quantization method. Source: [32].

introduced by Nutaro [33]. For solving a simple differential equation model by the quantization method, the simplest approach is to quantize the state space to be multiples of a fixed quantity D. Depending on the sign of $f(y,t)$, the next level of state may be higher or lower than the current state. The model event graph is shown in Figure 2.12. The next state value of y is the current state value with the quantity D in the sign of the current state value. Figure 2.11b shows the results achieved by the quantization method are much more satisfactory than the results from the Euler method. There is little deviation in behavior for different sized quanta D. All solutions converge quickly to the steady state value of 1.0, which is the correct value of the equation. However, in the Euler method approach, the results end up oscillating around the steady-state value. Even the closest solution has never been closer to the true value than the quantization method. This study illustrates the execution precision and efficiency of the DES structure.

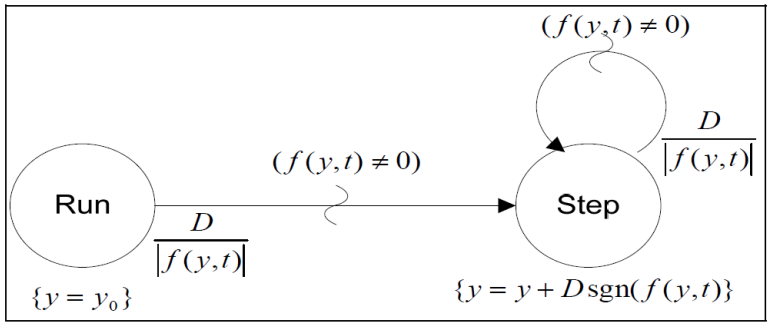


Figure 2.12. The event graph of DES structure quantized model. Source: [32].

2.2.5 Event-Driven Movers and Path Managers

Many simulation scenarios involve the location and movement of entities, and the location of entities plays a key role in the simulation scenario, such as a combat simulation. Conventionally, the entities movement simulations are mostly handled under the construction of the TS utilizing the fixed $\triangle t$ to calculate the next entity's location and simulation progress based on the previous time step $\triangle t_{-1}$ result. The result precision and the simulation execution efficiency in the TS structure simulation are highly dependent on the size of TS $\triangle t$ while the efficient event-driven moving entity structure can compensate for the drawback of the TS architecture. Buss et al. [15] propose a Listener Event Graph Object (LEGO) framework for Simkit to accomplish the moving entities scenario with even more complex interactive behavior between entities under DES architecture. The LEGO framework breaks the simulation into subclasses and each subclass has certain mutual events that get triggered when a relevant event transition happens during the simulation. There two subclasses: 1) Path Manager, and 2) Simple Mover, that manage the entity moving scenario under DES architecture. These two entities have mutual event listening pattern, which is shown in Figure 2.13. The events are heard and schedule mutually by the identical events.

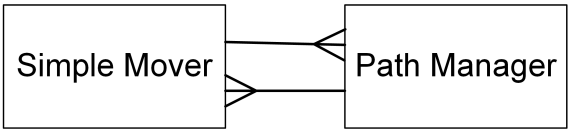


Figure 2.13. A DES event listening pattern of Path Manager entity and Simple Mover entity.

The event graphs of two entities, which manage the moving entities scenario are illustrated as follows:

- **Path Manager class:** The path manager class carries the designated waypoint and assigns a waypoint to the mover entities one at a time. Figure 2.14 shows the event graph of the "Path Manager" class. Each event gets scheduled during the simulation from other entities, and it is also be possible to schedule the event in other classes with the same signature during the simulation. When the simulation starts running the "Run" event is triggered and the first waypoint is passed to the "MoveTo" event in Point2D format. Point2D is the two dimensional coordinate system in Java. The waypoint will be passed to the mover entity.
- **Simple Mover class:** In Figure 2.15, the simple mover entity has two parameters: 1)

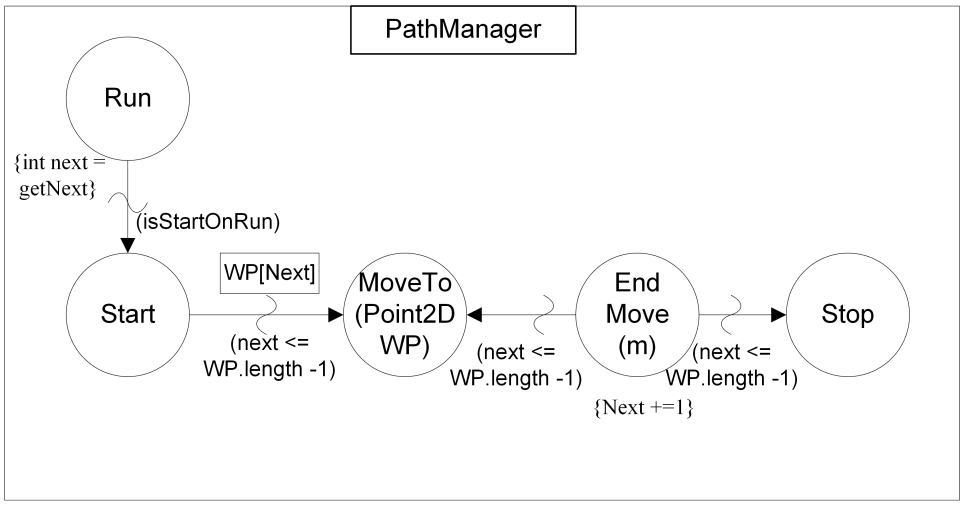


Figure 2.14. The DES event graph of path manager. This entity manages the next goal waypoint of mover entities.

max speed and 2) the initial location. Once the "MoveTo" event is scheduled by the "Path Manager" class, the next waypoint gets assigned and schedules the "StartMove" event in this class. The time T_1 for the mover to travel from the current position to the next waypoint is calculated by the $T_1 = \frac{Distance}{MaxSeed}$. The "EndMove" event will be schedule after T_1 to represent the end of the move by the mover from the first waypoint. If the "Stop" event is triggered before the mover reaches the waypoint, the "EndMove" event gets canceled at the same time, but the current waypiont will still be pending in the path manager.

The "EndMove" is scheduled when the mover reaches the next waypoint, and a new waypoint is pulled from the path manager class when the "Start" event is scheduled again. When the "EndMove" is scheduled by the "Stop" event from the mover class in the middle of the waypoint, the pending incomplete continues for the mover, and the manager does not provide a new one until the current waypoint is reached.

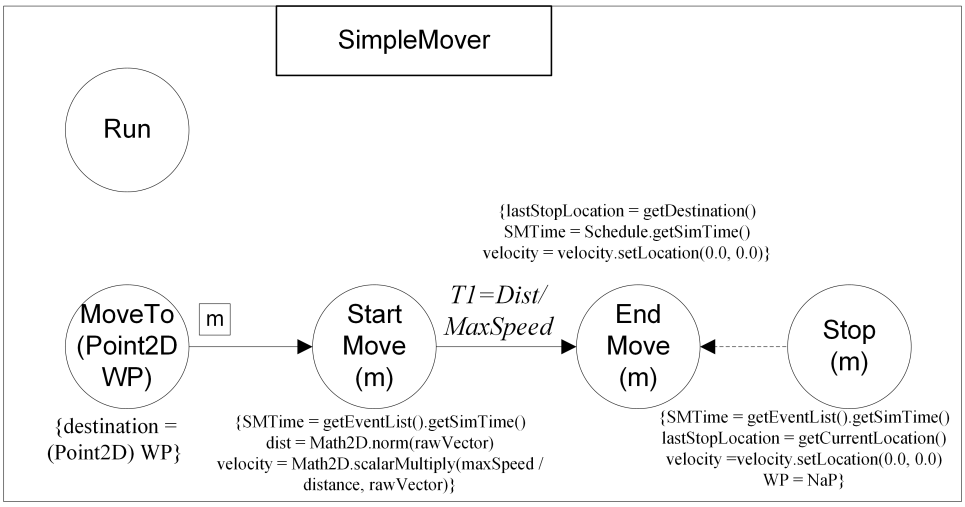


Figure 2.15. The DES event graph of mover entity. This entity represents the location of the object and receives the waypoints from the path manager.

2.2.6 Event-Driven Sensors and Targets

Extending from the DES mover and path manager structure, the dynamic sensors and targets simulation scenario is also constructed by the event-driven structure. DES sensor models can be executed efficiently and accurately in the simulation scenario because of their event-driven structure. There are no redundant computational procedures between scheduled events. There are two basic kinds of DES sensor models used in this work, which are introduced here to explain this concept:

Cookie-Cutter Sensor

The cookie-cutter sensor is the simplest sensor model. The detection algorithm is based on a detection perimeter of the sensor. The target detection mechanism is determined by three entity classes:

- **Sensor class.** The sensor entity maintains a list of contact targets for the further detection algorithm process. There are two events to manage the detection mechanism, and the sensor entity event graph is shown in Figure 2.16. When the "Detection" event happens, the sensor keeps the contact list of the target that is detected. When the "UnDetection" event is scheduled, the target stored in the contact list will be removed from the sensor entity. The events in the sensor entity do not have scheduling arcs to indicate the event flow. This is because all the events

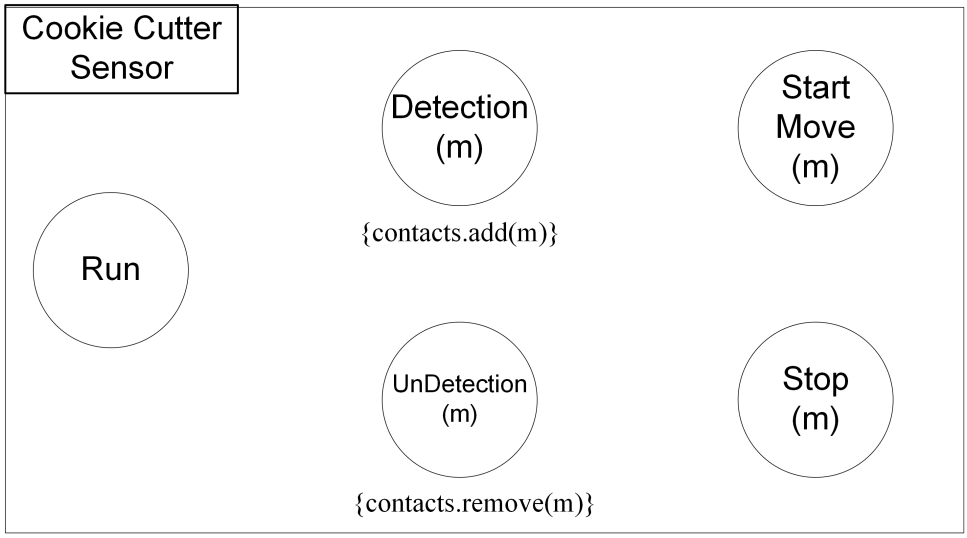


Figure 2.16. Cookie-cutter sensor event graph.

are scheduled by the events with the same signature from other entities with the event listener patterns. The event listener pattern manages the event invoking process by designating the event triggering direction and event name between two entities that contain mutual event activities during the simulation. The event with the same signature and linked by the event listener pattern gets scheduled when the associated event is scheduled in other entity.

- **Referee class.** The "Referee" entity event graph is shown in Figure 2.17. This entity determines if the "EnterRange" and "ExitRange" events are scheduled by listening to the location of the sensor and the target through the simulation when the "StartMove" event is scheduled. As long as a target remains outside the maximum detection range of a given sensor, there is no need to consider any interaction between the sensor and the target. The interruption of a scheduled event is represented by a dotted arc. When "Stop" event is triggered by the sensor or target entity, the scheduled "EnterRange" event gets canceled if the target has not reached the maximum detection range. while the "ExitRange" On the other hand, the "ExitRange" event gets canceled if it is scheduled after the "EnterRange" event happened.
- **Mediator class.** The "Referee" entity manages the contact event between two mover entities only. The detection condition after the movers make contact needs to be defined. Because different detection algorithms can be applied when the target and sensor make contact, another entity, which defines the detection algorithm of the

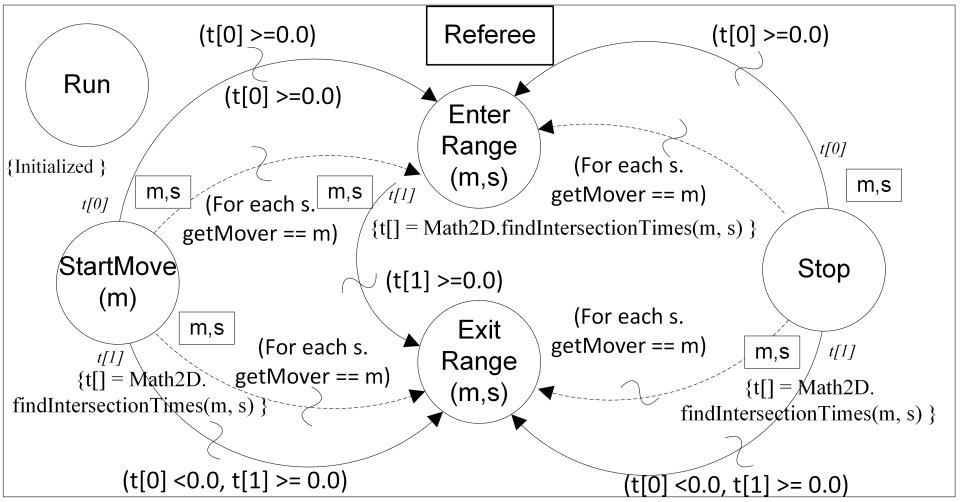


Figure 2.17. Referee event graph determines whether the movers enter the maximum detection range of the sensor.

sensor the target will be introduced. The "Mediator" is the entity that defines the detection algorithm after the contact event is announced by the "Referee". Figure 2.18 shows the event graph of the cookie-cutter sensor mediator. The mediator determines t_D, the time to schedule the "Detect" event, when the "EnterRange" event is scheduled. Since the detection happens at the same time as the "EnterRange" happens in the cookie-cutter model, the "Detect" event gets scheduled with time delay $t_D = 0$. The "UnDetect" event gets scheduled when the "ExitRange" event is scheduled by the Referee entity.

Figure 2.19 is an example scenario of a cookie-cutter sensor detecting a moving target. When simulation starts at t_{-1}, an aircraft flies from the orange diamond along the blue dotted line at speed $\mathbf{s}$. At simulation time t_0, an "EnterRange" event is calculated and scheduled by the referee, the target hits the detectable perimeter (green star) of the sensor and then the "Dectect" event is scheduled by the mediator entity. The sensor declares that the target was detected at t_0. The "UnDetect" event is scheduled by the referee at t_4 and the "UnDetect" event is scheduled by the mediator at the same time. The target was undetected when it reached the red square at t_4. The resulting event schedule is shown in Table 2.1.

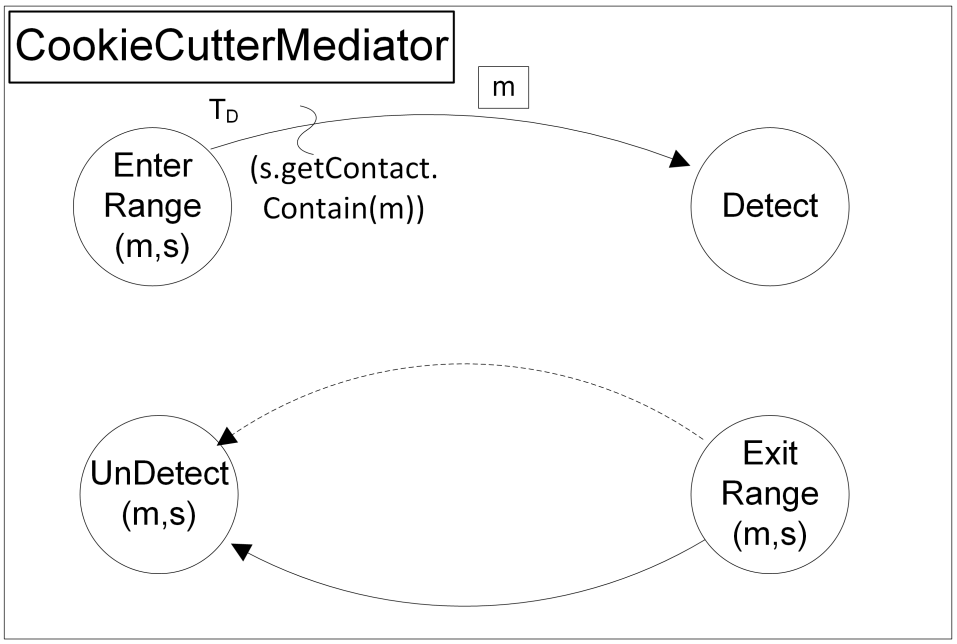


Figure 2.18. Mediator event graph ensures fair adjudication of whether detection occurs when the target enters the detectable range of the sensor. Adapted from [15].

Table 2.1. Cookie-cutter sensor simulation event schedule

Simulation time	Scheduled event
t_{-1}	Target approaches, not yet detectable
t_0	Target enters range
t_0	Target is detected
t_2	Target exits range, no longer detectable
t_2	Target is undetected

Constant-Rate Sensor

The cookie-cutter sensor model is an ideal case of a sensor model since the detection time delay of the target reaches the sensor maximum detection range, which is zero. In reality, there is always the processing latency and false detection rate that exist in the real radar system. To consider a TS detection behavior of a search radar sensor, as long as the target is within the maximum detection range (the yellow perimeter), the sensor has a constant P_d to detect the target in every time step $\triangle t$ untill the detection of target happens. The detection attempts are a sequence of Bernoulli trials with identical P_d. The number of

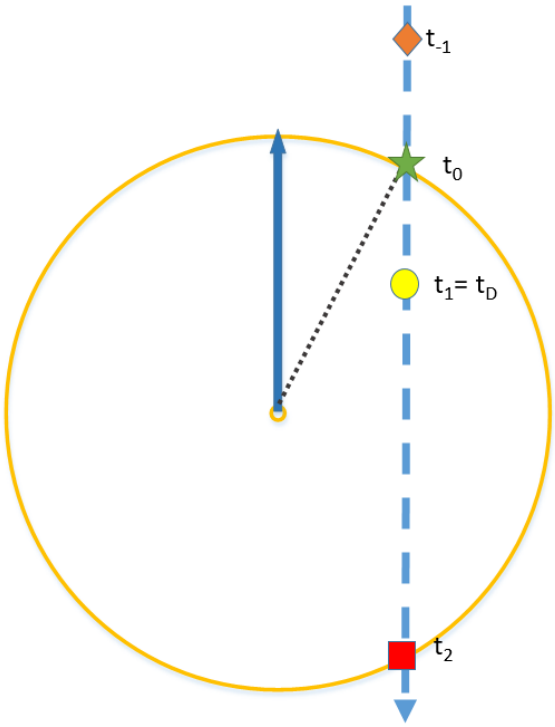


Figure 2.19. DES cookie-cutter and constant-rate sensors scenario.

steps to detect the target is a geometric random variable with parameter P_d, so the Mean Time To Detect (MTTD) is $\frac{\Delta t}{P_d}$. The TTD can be approximated by an exponential (μ) RV, where $\mu = \frac{\Delta t}{P_d}$ [34]. An exponential random variable with a MTTD = μ represents the TTD of a target after the target enters the detection perimeter (the green star in Figure 2.19). The TTD: $t_D \sim Exponential(\mu)$ is the time delay for the constant-rate sensor mediator to schedule the "Detect" event when the "EnterRange" event is scheduled by the referee. The TTD represents a sequence of Bernoulli trials N with identical probability p until the first successful detection while the $t_D = 0$ is the cookie-cutter mediator entity. The t_D brings stochastic time delay into the detection algorithm, which makes the content rate sensor more realistic than the cookie-cutter model. The resulting scenario event schedule is shown in Table 2.2.

The constant-rate sensor model represents a more realistic situation for a sensor to detect targets than the simple cookie-cutter sensor model. However, there are two points of ambiguity with this design:

Table 2.2. Constant-rate sensor simulation event schedule

Simulation time	Scheduled event
t_{-1}	Target approaches, not yet detectable
t_0	Target enters detectable range
$t_1 = t_0 + t_D$	Target is detected
t_2	Target exits range, no longer detectable
t_2	Target is undetected

- There is no clear expression to define the relationship between the MTTD of t_D and the desired parameters within radar equations.
- There is only one mean time-to-detect μ for the whole detectable area of a sensor. This assumption does not account for the RCS variation of a target at different angles, which are the typical scenarios in dynamic sensor-target simulations. The relative positions between targets and the sensor is varying as well as the aspect angle in between.

More-detailed radar parameters need to be considered in the sensor model with further analysis of sensor model performance. As shown later in this work, a hybrid approach allows the DES framework to handle these complexities.

2.3 Computational Radar Measurements

The radar scenario involves a transmitter and a receiver with the same or two different antennas, as well as a target and a signal that travels the round-trip between the radar and the target. In most cases, the transmitter and the receiver are at the same location, which is called a mono-static radar system, while the different locations of the transmitter and receiver is called bi-static radar. The transmitted signal is usually an electromagnetic signal that can be described by a carrier sine wave at frequency f_c with modulation schemes of amplitude, phase, and frequency. Radar system target detection is based on demodulating the scattered signal from the antenna to targets. The changes observed in the returned signal can provide information about the target position and motion. The time delay of the returned signal yields information on the range. The antenna pointing direction yielding maximum return strength provides the azimuth and elevation of the target relative to the

radar [35]. However, the gain slope of a conventional antenna beam peak is usually too flat to give accurate angle information. Special tracking techniques such as beam switching or monopulse are used for high accuracy angle measurements.

2.3.1 Radar Range Equation (RRE)

The RRE is used to compute the maximum detection range of the sensor to a target by determining the time delay of the returned signal. In radar, the relationship between the delay τ and the range R is given by

$$R = \frac{1}{2}\tau C$$

where C is the velocity of propagation, which is equal to the speed of light in a real atmosphere. The factor of $\frac{1}{2}$ is the result of the round trip travel time of the signal.

The maximum detection range (Rmax) is derived by the RRE as follows [36]:

$$R_{max} = \sqrt[4]{\frac{P_t \cdot G_t \cdot G_r \cdot \lambda^2 \cdot \sigma_T \cdot N_{Int}}{4\pi^3 \cdot K \cdot T_0 \cdot B_r \cdot F_R \cdot (S/N)_1 \cdot \chi^2_{df=4}}} \tag{2.3}$$

and maximum Unambiguous range (Ru) of the sensor due to the radar prf constraint is derived as follows:

$$R_u = \frac{C}{2 \cdot prf} \tag{2.4}$$

The detectable range of targets from the sensor has to satisfy both Equations 2.3 and 2.4 i.e., the threshold range R_{detect} in which the sensor will detect targets needs to meet these two criteria:

$$R_{detect} < R_u \text{ and } R_{detect} < R_{max} \tag{2.5}$$

These criteria are true of a low prf radar that does not have range ambiguities. It is possible to operate in higher prf conditions and resolve the range ambiguities in the processing.

The parameters applied in the RRE have been classified into two categories in this study: 1) the parameters that can only be configured by the radar system and under the control of the radar designer. These parameters are listed in Figure 2.20, and they can be set up as in the modeler's attempt; and 2) the rest of the parameters that have an interactive relationship with

Ave Transmitter power	Tx Antenna Gain	Rx Antenna Gain	Signal Wave Length	Boltzmann's constant
Pt	Gt	Gr	λ	K
1e5 (W)	1e3 (W)	1e3(W)	0.0375 (m)	1.38e(-23)
Standard Temperature	Receiver Bandwidth	Receiver Noise Figure	Min Signal to Noise Ratio by Single Pulse	
To	Br	Fr	(S/N)1	
290 (K)	1e6 (Hz)	10	15 (dB)	

Figure 2.20. Radar range equation parameters only configured by sensors.

other factors in the scenarios are set up accordingly. The target's RCS is the parameter that has corresponding responses with the aspect angle between the target and the sensor [37]. The RCS value varies as the aspect angle of the targets changes. This is a dominant factor that can have significant effect on the RRE in a dynamic targets and sensors simulation scenario, because the locations and relative aspects of targets and sensors are subject to change as the simulation trajectories are executed.

Radar Cross Section (RCS) σ_T

RCS is the property of a scattering object that represents the magnitude of the echo signal returned to the radar by the target. Electromagnetic waves are normally diffracted or scattered in all directions when incident on a target. The intensity of the backscattered energy with the same polarization as the radar's receiving antenna is used to define the target RCS. When a target is illuminated by RF signal, it acts like an antenna and backscatters RF energy. Waves reflected and measured in the near field are spherical, while in the far field, the wavefronts are decomposed into a linear combination of plane waves. When the signal operational wavelength is much smaller than the target extent, this condition is referred to as the optics region. The optics region is applied in the scenario in this research.

Assume the power density of a wave incident on a target located at range R away from the radar is P_{Di}. The amount of reflected power from the target is [38]

$$P_r = \sigma_T P_{Di} \tag{2.6}$$

where σ_T denotes the target RCS. The power density of the backscattered signal at the receiving antenna is as follows:

$$P_{Dr} = \frac{P_r}{4\pi R^2} \tag{2.7}$$

Equating Equation 2.6 and 2.7, yields the RCS:

$$\sigma_T = 4\pi R^2 \left(\frac{P_{Dr}}{P_{Di}}\right) \tag{2.8}$$

RCS is the measure of area in the unit of m^2 and fluctuates as a function of radar aspect angle and frequency.

Perhaps the simplest RCS target is the measure of a sphere. Due to the identical geometry of a sphere from any angle, the RCS is independent to the aspect angle. A constant RCS value is shown in Figure 2.21 for the aspect angle of the sphere $0° \leq \phi \leq 360°$. The RCS of a sphere is defined as follows [39]:

$$\sigma_T = \frac{2\pi a}{\lambda} \tag{2.9}$$

where the radius of sphere is a, the signal wavelength is λ. The measure is in optical region, which $\frac{2\pi a}{\lambda} \gg 1$.

However, when two identical spheres are placed apart by λ (the left of Figure 2.22) and 2λ the RCS (the right of Figure 2.22) can vary considerably depending on view aspect and signal frequency. The RCS model is defined as follows:

$$\frac{\sigma_r}{\sigma_T} = 2\left[1 + \cos\left(\frac{4\pi l}{\lambda}\sin\phi\right)\right] \tag{2.10}$$

where l is the separation and ϕ is the aspect angle to the two spheres with $\theta = 90°$. Although this is a rather minimalist example of a "simple" complex target, its highly variable behavior is indicative of more complicated targets.

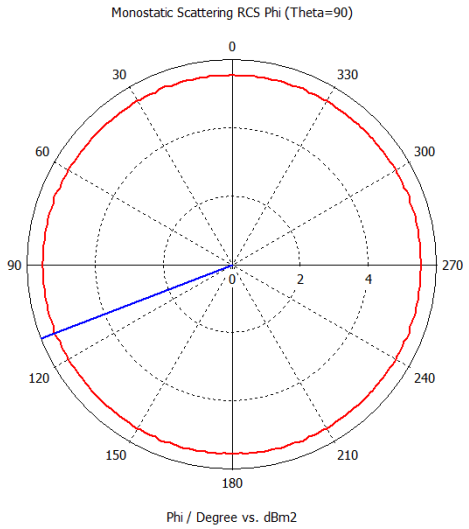


Figure 2.21. A sphere object, the simplest geometry of RCS measure, in azimuth angle $0° \leq \phi \leq 360°$.

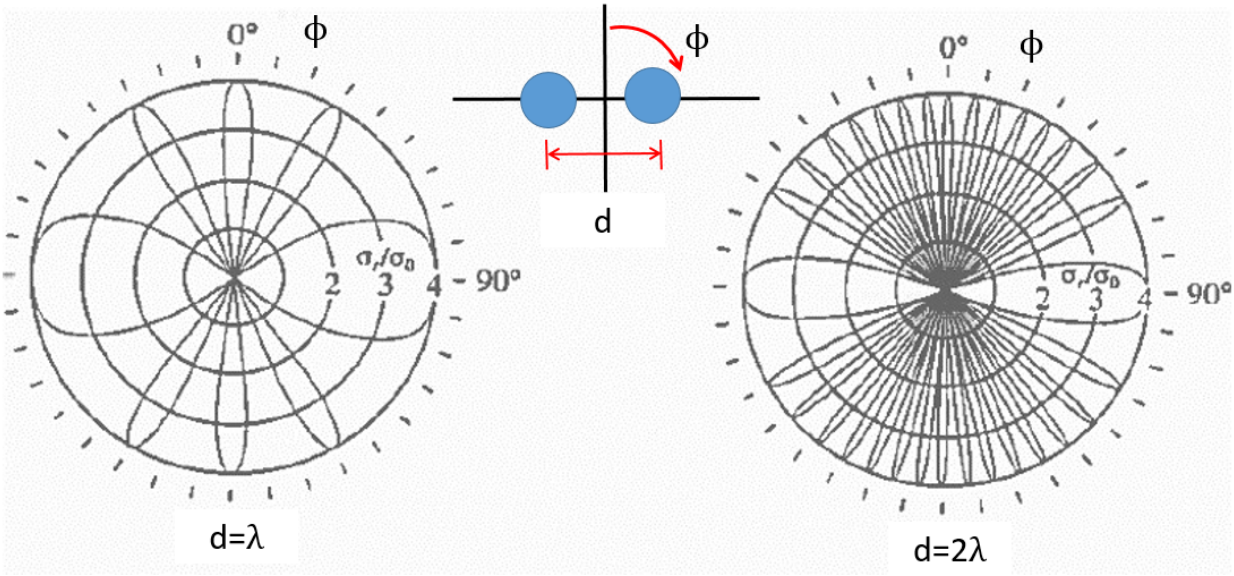


Figure 2.22. The RCS measure of two sphere objects are set apart by λ (left) and 2λ (right) , in azimuth angle $0° \leq \phi \leq 360°$. Adapted from: [40]

A further question is whether the physical size of the target indicates RCS size. An illustrative RCS example by comparing two different geometrical objects is shown in Figure 2.23. (a) is a flat plate object with the face perpendicular the to radio source, while (b) is a cone-Sphere object with the tip pointing to the signal source. Both objects have the same

$1m^2$ physical projecting area from the view of the signal source ($\phi = 0°$). The RCS models of these two objects in $\phi = 0°$ are computed as follows [41]:

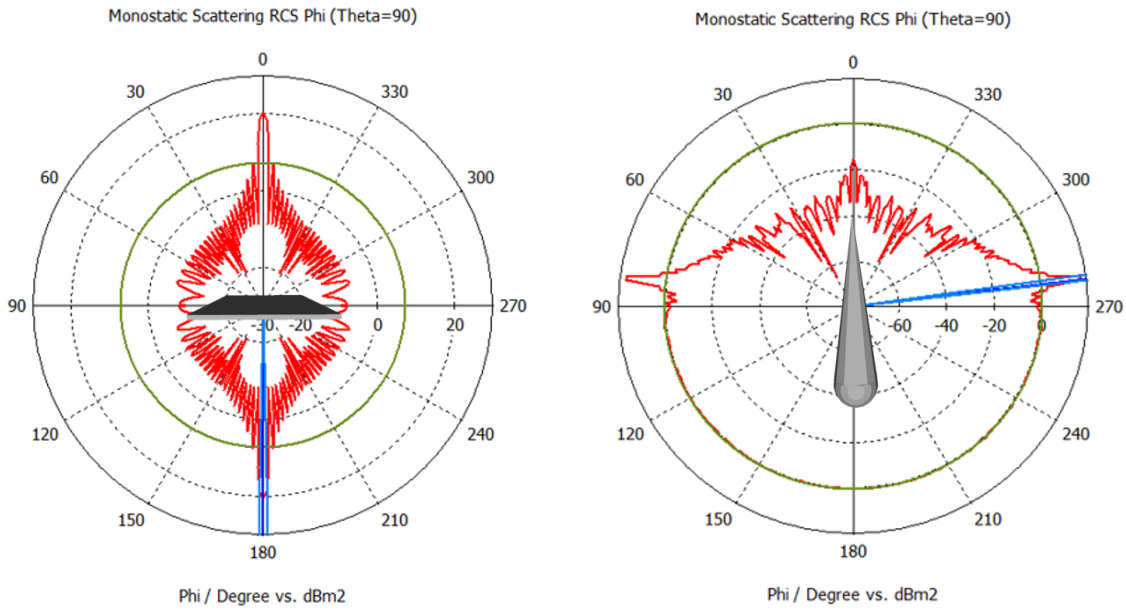
- The RCS of a flat plate at broadside, A is the area of plate

$$\sigma_{Plate} = \left.\frac{4\pi A^2}{\lambda}\right|_{\lambda=0.1m, f=3GHz, A=1m^2} = 1000(m^2)$$

- The RCS of a cone-sphere, a is the radius of the sphere with 30° of the cone angle

$$\sigma_{Cone} = \left.\frac{2\pi a}{\lambda}\right|_{\lambda=0.1m, f=3GHz, a=0.564m} = 0.001(m^2)$$

There is a million-to-one difference in the RCS of two targets even if each of them has the same projected area. Observing the examples of spheres and the two preceding objects, the geometry of the objects is definitely a critical factor of RCS.



(a) The RCS of a flat plate object (b) The RCS of a cone-sphere object

Figure 2.23. The RCS measurement of two objects with the same 1 m^2 physical projecting area at broadside, showing a million-to one difference in response characteristics.

The RCS of complex targets such as aircraft, missiles, vessels, and terrain can also vary

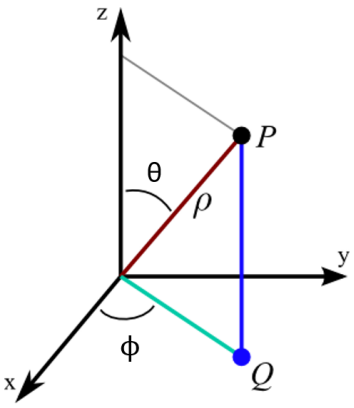


Figure 2.24. Definition of local spherical coordinate system. The point P is represented by the distance ρ with the elevation angle $0° \leq \theta \leq 180°$ and the azimuth angle $0° \leq \phi \leq 360°$. Adapted from [42].

considerably depend on the view aspect and radio frequency. In the left of Figure 2.25, a F-16 Falcon fighter model, which contains 2,056 vertices and 4,092 faces, is aligned along the x-axis facing positive x vector direction while the body and wings are on the $x - y$ plane herizontially. In this study, the RCS values are computed along the azimuth angle in a range of $0° \leq \phi \leq 360°$ while the aircraft is aligned on the $x - y$ plane as $\theta = 90°$ in a spherical coordinates system. The definition of a spherical coordinates in shown in Figure 2.24. Spherical coordinates determine the position of a point P in three-dimensional space-based on the distance ρ from the origin with the azimuth angle ϕ in a range of $0°$ to $360°$ on the $x - y$ plane and the elevation angle θ in a range of $0°$ to $180°$ from the z-axis.

The RCS of an F-16 falcon fighter 3D model is computed by a high-fidelity electromagnetic simulation software, CST Studio [43], in full-scale size using signal frequency 8 GHz. The entire aircraft is modeled as Perfect Electric Conductor (PEC) material. The RCS result is shown in the right of Figure 2.25. The RCS of the aircraft has agile change with the variance of the azimuth angle ϕ. There are large RCS at front, back, and sides of the aircraft while the RCS is relatively small at the diagonal angle of the model. Computing one angle of RCS from the target takes around one minute. A Two dimensional (2D) RCS elevation cut ($\theta = 90°$) with $0.5°$ resolution has $360 \times 2 = 720$ total angle of computation. It may take a personal computer one day to accomplish a batch. In the larger physical size model or when the model contains more vertices and faces, the computational time can be much longer. For a 3D RCS pattern with $1°$ resolution and $0.5°$ azimuth resolution, the total number of

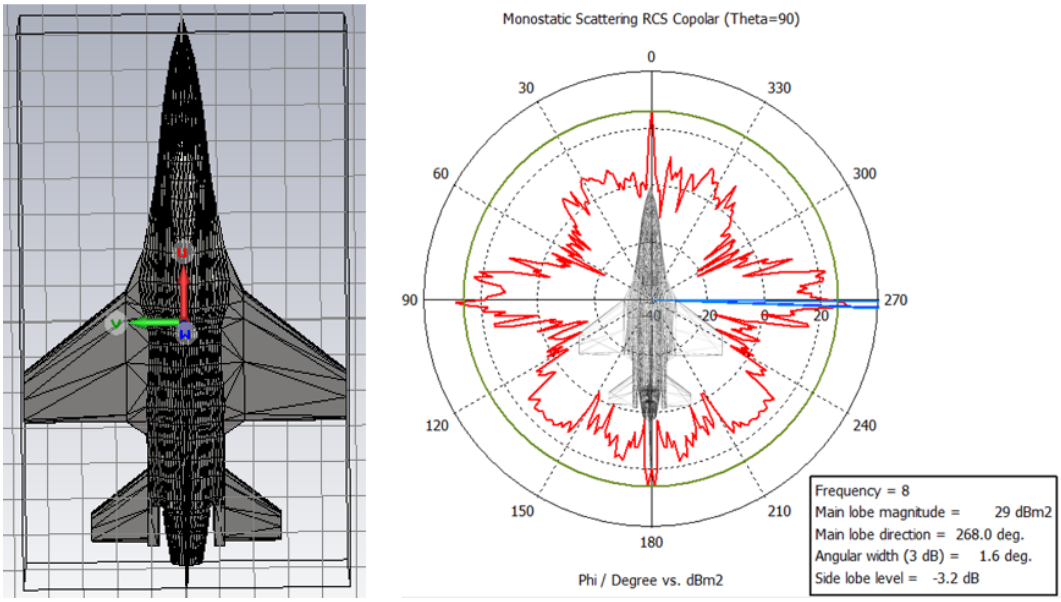


Figure 2.25. The RCS measure of F-16 Falcon fighter model in $\theta = 90°$ and $0° \leq \phi \leq 360°$. The F-16 model contains 2,056 vertices and 4,092 faces. See Appendix B for additional 3D and RCS models.

new rays, each computationally intense, is $180 \times 360 \times 2 = 129{,}600$. The computation time for a computer to complete a batch is unbearable.

This computational complexity can cause tremendous latency during the simulation if this atemporal factor, the RCS response of the object, is not isolated from others. To avoid latency, the RCS of the targets is precalculated for the simulation application in this research. Also the high-complexity object RCS models are performed by the HPC at the Naval Postgraduate School (NPS) [44]. With the enormous computational power provided by this resource, an excellent source of RCS models from arbitrary objects become available for this study.

Pulse Integration Improvement Factor N_{Int}

For a pulse radar, the signal is transmitted in a series of pulse trains. The scattering signal returning from the object contains multiple pulses during each antenna scan. The more the pulses returning from the target during each scan, the higher is the signal processing gain. The number of pulses illuminating the target and returning back to the receiver during each scan is called the pulse integration improvement factor and it is denoted as N_{Int} in Equation

2.3. The *pulse integration improvement factor* is determined by three radar parameters [45]:

- The radar prf (Hz): This factor represents the frequency of the radar that transmits the pulses in the pulse train. The $pri = \frac{1}{prf}$ is also commonly used and is thus called the pulse repetition interval (pri).
- The antenna 3dB beamwidth θ_{3dB}: This factor represents the effective beamwidth of the antenna to illuminate on the targets. The effective beamwidth is twice the angle from the maximum antenna gain to the half of the maximum antenna gain (3 dB decay). Only the azimuth angle effect is considered in the study of this 2D planar search radar model, though further fidelity might be gained by considering elevation angle effects.
- The antenna revolutions per minute (rpm): This factor defines the speed of the antenna sweeping mechanism.

There are two steps for computing the N_{Int}:

1. The effective Time on Target (ToT): The effective radar signal illumination ToT per revolution is calculated as follows:

$$ToT = \frac{60s}{rpm \times 360^{\circ}} \times \theta_{3dB\_AZ}\ (s) \tag{2.11}$$

2. The number of effective pulses provided to the integration improvement factor is calculated as:

$$N_{Int} = ToT \times prf \tag{2.12}$$

A mono-static radar assumption is applied in this search radar model scenario, which means the transmitting and receiving signal are at the same station. A coherent integration improvement factor is applied in this system and it is denoted as N_{Int} in Equation 2.3.

Target Fluctuation: Swerling III model $\chi^2_{df=4}$

The target detection algorithm utilizing the square law detector assumes a constant RCS (nonfluctuating target). This work was extended by Peter Swerling to consider four distinct cases of target RCS fluctuation. These cases have come to be known as Swerling models. Target fluctuation introduces an additional loss factor as compared to the case where fluctuation is not present [46]. Rather than being physically based, these stochastic variations

attempt to characterize complex fluctuation effects between the target RCS and radar pulses during a scan. The four cases of Swerling model approximations are defined as follows [47] :

1. **Case I:** The echo pulses received from a target on any one scan are of constant amplitude throughout the entire scan but are independent from scan to scan. It is also known as *slow fluctuation*. The target RCS amplitude varies according to a Chi-Squared probability density function with two degrees of freedom.
2. **Case II:** The fluctuations are independent from pulse to pulse rather than from scan to scan as in Case I. This is sometimes called *fast fluctuations.* The target RCS amplitude varies according to a Chi-Squared probability density function with two degrees of freedom.
3. **Case III:** The RCS is assumed to be constant within a scan and independent from scan to scan and fluctuates by a Chi-Squared probability density function with four degrees of freedom.
4. **Case IV:** The fluctuation is pulse to pulse and fluctuates by a Chi-Squared probability density function with four degrees of freedom.

Swerling showed that the statistics associated with case I and II models apply to targets consisting of many small scatters of comparable RCS values, while the statistics associated with case III and IV models apply to targets consisting of one large RCS scatterer and many small equal RCS scatterers. Noncoherent integration can be applied to all four case models while coherent integration cannot be used when the target fluctuation is either case II or IV. This is because the RCS is not assumed to be constant between pulses. Swerling case III matches the application and is applied in this study. A Chi-Squared probability density function with four degrees of freedom is applied as the fluctuation loss in this study. This factor models the fluctuation loss from targets between each antenna scan. The RCS is assumed to be constant within a scan and independent from scan to scan; these conditions are the same as those in this simulation scenario. The sensor model generates stochastic time-to-detect results due to this fluctuation loss model which, is represented by a Chi-Squared random variable with degree of freedom = 4. This Swerling loss factor is denoted as $\chi^2_{df=4}$ in Equation 2.3. Figure 2.26 presents the Rmax of the sensor to detect the target in one RCS by 50 replications with the fluctuation loss model of Swerling case III. This scenario represents the Rmax of the sensor to detect the stationary target in 50 scans.

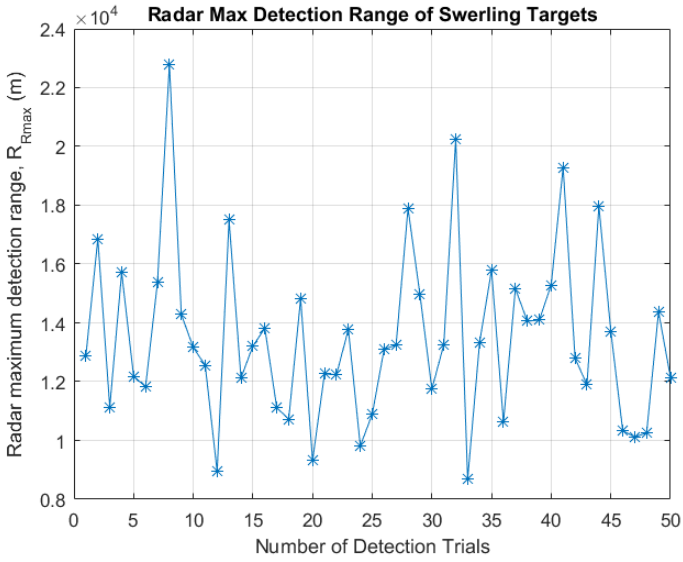


Figure 2.26. The maximum detection range of the radar to one target RCS with the fluctuation loss presented by case III Swerling model. The simulation is executed with 50 replications.

Due to the fluctuation loss model factor $\chi^2_{df=4}$, the Rmax is less than the model without fluctuation loss with the stochastic result at each antenna scan.

CHAPTER 3:
Prototype Advanced Sensor Models

The perfect is the enemy of the good.

–Orlando Pescetti [48]

3.1 Model I: Two-Tiers Sensor Model

The constant-rate sensor model in section 2.2.6 applies one MTTD = μ to represent the time delay TTD = t_D of the targets to the sensor once the targets enter the Ru. The detection attempts are a sequence of Bernoulli trials with identical probability P_d in number N of detection attempts. Because P_d varies as the relative position between the targets and the sensor changes, the constant-rate sensor applies only one μ to represent the t_D for all aspect angles. Section 2.3 illustrates that Rmax varies along with different aspect angles and distances between targets and the sensor. These changes result from the variation of RCS and the signal attenuation by the traveling distance, which affects Signal-to-Noise-Ratio (SNR) in the RRE in Equation 2.3. The sensor detection ability is affected significantly as the aspect angle and target distance change. The one μ structure fails to represent the aspect-angle variation effect as varying P_d in a dynamic target-sensor scenario.

3.1.1 Model Structure

In order to represent different P_d in a dynamic scenario, a two-tier sensor model is proposed that splits the maximum detectable range into two MTTD: μ_1 and μ_2 for two exponential random variables to represent two different TTD: t_{D1} and t_{D2} when targets are in different positions of the detection region: $t_{D1} \sim Exponential(\mu_1)$, $t_{D2} \sim Exponential(\mu_2)$. Figure 3.1 displays the event graph of a two-tier constant-rate mediator. When the "EnterRange" event is triggered by the referee, the time for the target traveling to the midpoint towards its waypoint t_m is calculated and applied for the condition decision that if the $t_D > t_m$ with the $t_D = TTD_1$. If the condition is true, the "EnterMidPoint" event gets scheduled in t_m. Then the second attempt of "Detect" event gets scheduled at $t_D = TTD_2$. As the similar condition in the constant-rate mediator, the "UnDetect" event gets scheduled once the "ExitRange" event is scheduled by the referee entity or the "UnDetect" event is canceled if the "Detect" event is scheduled after the "ExitRange" event occurs.

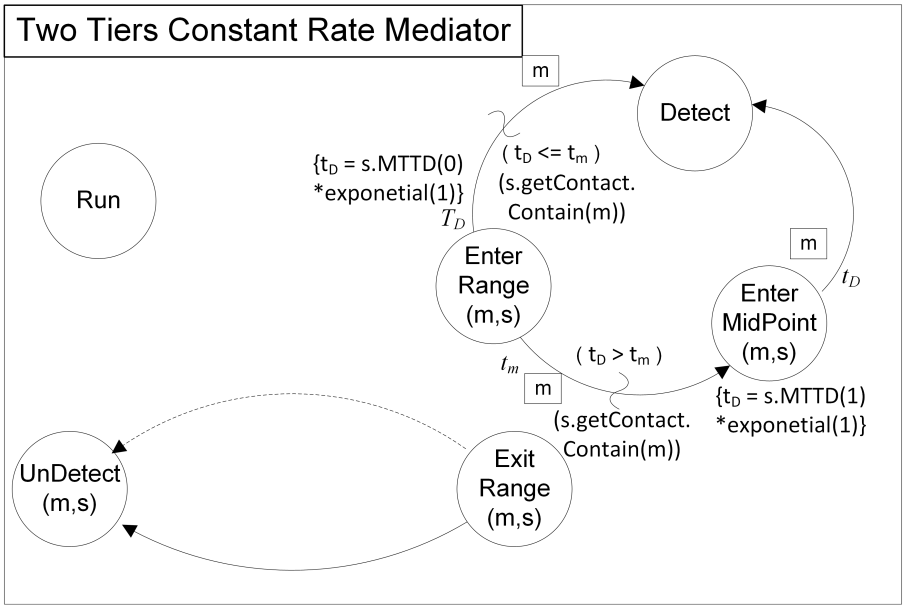


Figure 3.1. The two-tier TTD mediator event graph. The mediator ensures fair adjudication of whether detection occurs when the targets enters the detectable range of the sensor, providing two possible TTD RVs in two detection regions.

The following example scenarios explain the behavior of the two-tier constant-rate sensor. The two MTTDs' transition boundary is located at the midpoint of the targets' waypoint, which is shown in the purple circle of Figure 3.2. Therefore, three possible simulation scenarios exist for this sensor:

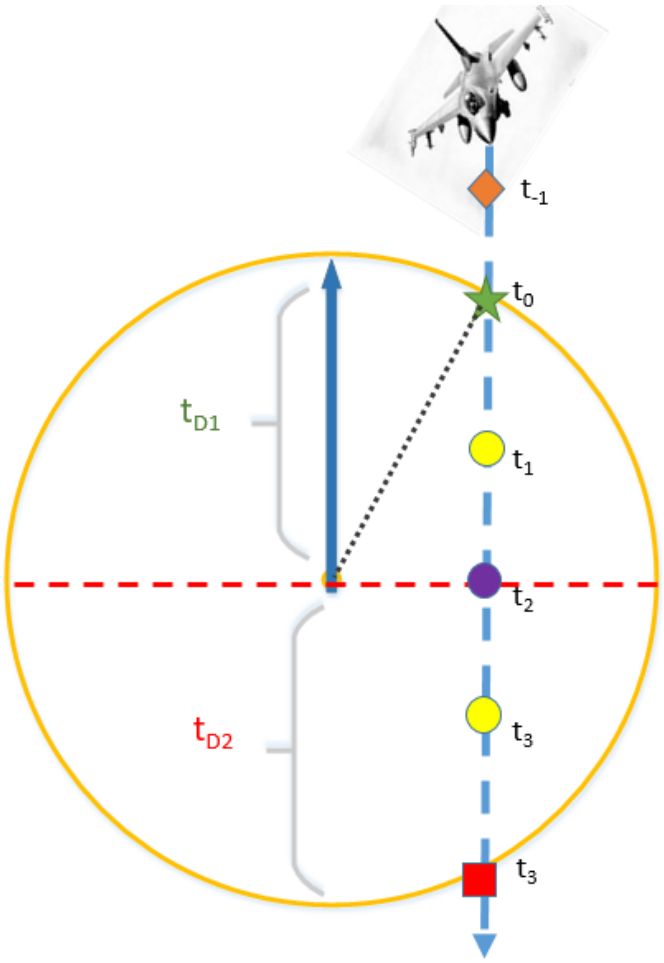


Figure 3.2. A two-tier constant-rate DES sensor operation scenario.

3.1.2 Scenario I: Detection Occurs in Region 1

The target starts approaching the sensor at t_{-1} on the orange diamond of Figure 3.2. It is similar to the constant-rate sensor in the section 2.2.6. The target enters the sensor maximum detection range at t_0 (green start) and the detection event is scheduled at $t_1 = t_0 + t_{D1}$ (yellow circle) in Figure 3.2, which happens before the target enters the bottom region 2 with different MTTD. The t_{D1} is an exponential RV with $\text{MTTD}_1 = \mu_1$ and it is denoted as $t_{D1} \sim Exponential(\mu_1)$. The overall event schedule is shown in Table 3.1.

Table 3.1. Simulation event schedule for two-tier constant-rate sensor detects at region 1

Simulation time	Scheduled event
t_{-1}	Target approach, not yet detectable
t_0	Target enters detectable range 1
$t_1 = t_0 + t_{D1}$	Target is detected
t_4	Target exits range
t_4	Target is undetected

3.1.3 Scenario II: Detection Does Not Occur in Region 1

Because the t_{D1} is an exponential RV, the first scheduled detection event time t_1 in region 1 may be later than the target arrives at the center of its waypoint at t_2 (the purple circle) in Figure 3.2. In this case, the detection event t_1 gets canceled and the other detection event gets scheduled when the target enters the middle of the waypoint at t_2. The second target detection time $t_3 = t_2 + t_{D2}$. t_{D2} is an exponential RV with $\text{MTTD}_2 = \mu_2$. The t_{D2} is to represent the time delay when the target passes through the middle of the waypoint and enters region 2, which is shown in the purple circle. It is denoted as $t_{D2} \sim Exponential(\mu_2)$. The overall event schedule is shown in Table 3.2.

3.1.4 Scenario III: Detection Does Not Occur in Region 1 or 2

A similar scenario as no detection happening in region 1 in section 3.1.3. In this scenario, the first scheduled detection time t_1 is longer than the time the target passes through the region 1 and the second detection t_3 scheduled from t_2 is also longer than the time when the target exits the maximum detection range at t_4 (the red square in Figure 3.2). In this case, the detection event t_3 gets canceled and scheduled as the target exits the maximum detection range at t_4 (red square). The overall event schedule is shown in Table 3.2, and the detection event has never happened through the scenario.

3.1.5 Regression Analysis

A simple linear-regression analysis is applied to both the constant-rate sensor model and the two-tier constant-rate sensor to analyze the significant factors that might affect the detection capability of the sensor models. The regression analysis is based on the observations, which

Table 3.2. Simulation event schedule for two-tier constant-rate sensor does not detect the target at region 1

Simulation time	Scheduled event
t_{-1}	Target approaches, not yet detectable
t_0	Target enters detectable range 1
$t_1 = t_0 + t_{D1}$	Target is detected after t_{D1}
if $t_1 = t_0 + t_{D1} > t_2$	Target is not detected at region 1. Cancel target detection at t_1
t_2	Target enters detectable range 2
$t_3 = t_2 + t_{D2}$	Target is detected in region 2
t_4	Target exits range
t_4	Target is undetected

Table 3.3. Simulation event schedule for when two-tier constant-rate sensor does not detect the target in region 1 nor in region 2

Simulation time	Scheduled event
t_{-1}	Target approach, not yet detectable
t_0	Target enters detectable range
$t_1 = t_0 + t_{D1}$	Target is detected after t_{D1}
if $t_1 = t_0 + t_{D1} > t_2$	Target is not detected at region 1. Cancel target detection at t_1
t_2	Target enters detectable range 2
$t_3 = t_2 + t_{D2}$	Target is detected in region 2
t_4	Target exits range
t_4	Target is undetected

are generated in the 65 DPs by the NOLH design. The DoE assigned a range of values for the parameters of interest in the sensor models. The parameters of constant-rate sensor are:

- **MTTD1:** The MTTD of the sensor to detect the target.
- **speed:** The maximum speed of the target.
- **IncidentAngle:** The incident angle of the target flies close to the sensor.

The parameters of the two-tier sensor model are:

- **MTTD1:** The MTTD of the sensor to detect the target in the detection region 1.
- **MTTD2:** The MTTD of the sensor to detect the target in the detection region 2.

- **speed:** The maximum speed of the target.
- **IncidentAngle:** The incident angle of the target flies close to the sensor.

Each DoE is executed with 100 replications for analyzing the stochastic behavior of these two sensor models. This yields a total number of 65 × 100 = 6,500 data points as the observations for the linear-regression analysis. Both meta-models are fit with the factorial and polynomial interaction predictors up to a degree of 2. The detailed DoE and regression analysis methodology are introduced in section 4 and section 5.

Figure 3.3 shows the goodness of fit of the both constant-rate sensor and the two-tier constant-rate sensor to the regression meta-models. The RSquared values are 0.2375 and 0.27452 respectively. The RSquared value of both model do not show a good fit by the meta-model in this case. Nevertheless, the linear-regression analysis allows the simulator to determine the significant factors from the complex and unknown models.

3.1.6 Apply Gamma RVs as TTD

The linear-regression analysis results from both the one constant-rate sensor and two-tier constant-rate sensor models do not show a good fit and low estimation error rate on the meta-models due to the low RSquared values. This indicates that the linear-regression analysis does not create a good model for predicting TTD of the constant-rate sensors. In the next approach, the sensor construction is similar to the previous constant-rate sensors except the original TTD exponential variables are being replaced by gamma RVs. The definition of gamma RV is as follows [49]:

$$gamma(\alpha, \beta) = f(y) = \begin{cases} \frac{y^{\lambda-1} e^{\frac{-y}{\beta}}}{\beta^{\alpha}\Gamma(\alpha)}, & 0 \leq y < \infty, \\ 0, & \text{elsewhere,} \end{cases} \tag{3.1}$$

where

$$\Gamma(\alpha) = \int_0^{\infty} y^{\alpha-1} e^{-1} dy.$$

α is the parameter of gamma RV shape and β is the scaling parameter of the gamma-distributed random variable, which is a positive constant. Figure 3.4 displays a family of different shapes and scaling parameters for gamma RV Probability Density Function (PDF). The gamma RV family has a high skewness RV, such as $gamma(1, 1)$ (red

curve) and low skewness RV (blue curve), such as $gamma(3, 1)$.

Exponential RVs are a special case of $gamma(1, \beta)$ RVs, which are the high skewness branch of gamma RVs. In this approach, the lower skewness TTD $\sim gamma(3, 2)$ is chosen for the constant-rate sensor. The linear-regression analysis results for the one constant-rate sensors and two-tier constant-rate sensors to predict TTD of the targets are as follows:

- One constant-rate sensor with TTD $\sim gamma(3, 2)$:
  The linear-regression analysis is to predict TTD based on the $65 \times 100 = 6500$ simulation data. The simulation is conducted by 65 DPs by NOLH DoE with two predictor parameters: 1) target incident angle, 2) target speed. Each DP executes 100 replications. The RSquared value of meta-model increases from 0.237527 to 0.739756. The results show that the TTD $\sim gamma(3, 2)$ sensor model has a lower prediction error rate in the meta-model.
- Two-tier constant-rate sensor with $TTD_1, TTD_2 \sim gamma(3, 2)$:
  The same DoE is applied to the two-tier constant-rate sensor with gamma RV. The linear regression meta-model for predicting variable TTD by 6500 observation points has RSquared value increases from 0.26185 to 0.704019. The meta-model also has a lower prediction error rate with a lower-skewness gamma RV to represent TTD for the constant-rate sensors.

The gamma RV constant-rate sensors do not represent or imply any specific meaning of the sensor-detection algorithm in this moment. However, the regression analysis for studying the behavior of an unknown system, such as the constant-rate sensor models, can create the meta-model that have certain predictability to the baseline sensor models.

Response timeToDetect

Effect Summary

Source	LogWorth
MTTD1	37.041
speed	17.522
MTTD1*speed	11.532
IncidentAngle	7.111
MTTD1*IncidentAngle	5.307
MTTD1*MTTD1	4.303
IncidentAngle*IncidentAngle	3.189
speed*IncidentAngle	0.447
speed*speed	0.119

Summary of Fit

RSquare	0.237527
RSquare Adj	0.233927
Root Mean Square Error	0.735604
Mean of Response	0.791793
Observations (or Sum Wgts)	1916

(a) constant-rate sensor model regression with predictors to degree of 2

Response timeToDetect

Effect Summary

Source	LogWorth
MTTD1	15.381
speed	5.867
IncidentAngle 2*IncidentAngle 2	5.107
MTTD1*MTTD2	5.011
speed*IncidentAngle 2	4.934
MTTD2*IncidentAngle 2	4.799
IncidentAngle 2	3.842
MTTD1*IncidentAngle 2	3.767
MTTD1*MTTD1	3.692
speed*speed	3.668
MTTD1*speed	3.413
MTTD2	3.362
MTTD2*speed	3.004
MTTD2*MTTD2	0.363

Summary of Fit

RSquare	0.274528
RSquare Adj	0.269638
Root Mean Square Error	0.711627
Mean of Response	0.837691
Observations (or Sum Wgts)	2092

(b) Two-tier constant-rate sensor model regression with predictors to degree of 2

Figure 3.3. The regression analysis result: (a) the constant-rate sensor model, (b) the two-tier constant-rate sensor model. The regression observations are executed by the 65 DPs of NOLH design with 100 replications of each DoE. The predicted time to detect of meta-model with the predictor parameters in factorial and polynomial interaction terms in degree of 2.

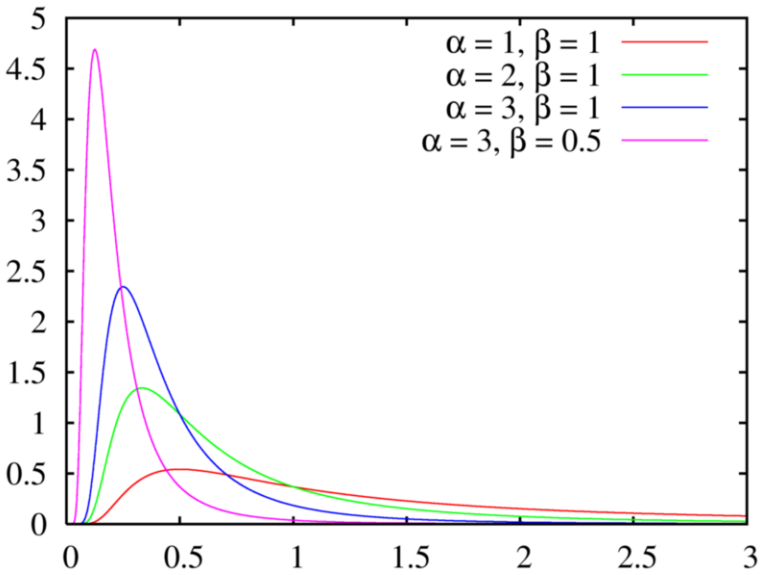


Figure 3.4. Gamma random variable PDF of different values of shape parameter α and scaling parameter β. Source: [50]

3.2 Model II: Fractional MTTD Region Structure

The constant-rate sensors with one or two MTTD RVs are still insufficient to represent the range and aspect angle effects derived from the RRE, which is introduced in section.2.3. Based on the RRE, the aspect angle and range between the target and sensor take an important part in determining the detection ability of the sensor to the targets in that specific geometry location during the simulation. Therefore, two advanced structures for constant-rate sensors are proposed to cover the information of aspect angle and range from the target.

3.2.1 Multiple Fan-Sectors MTTDs Sensor

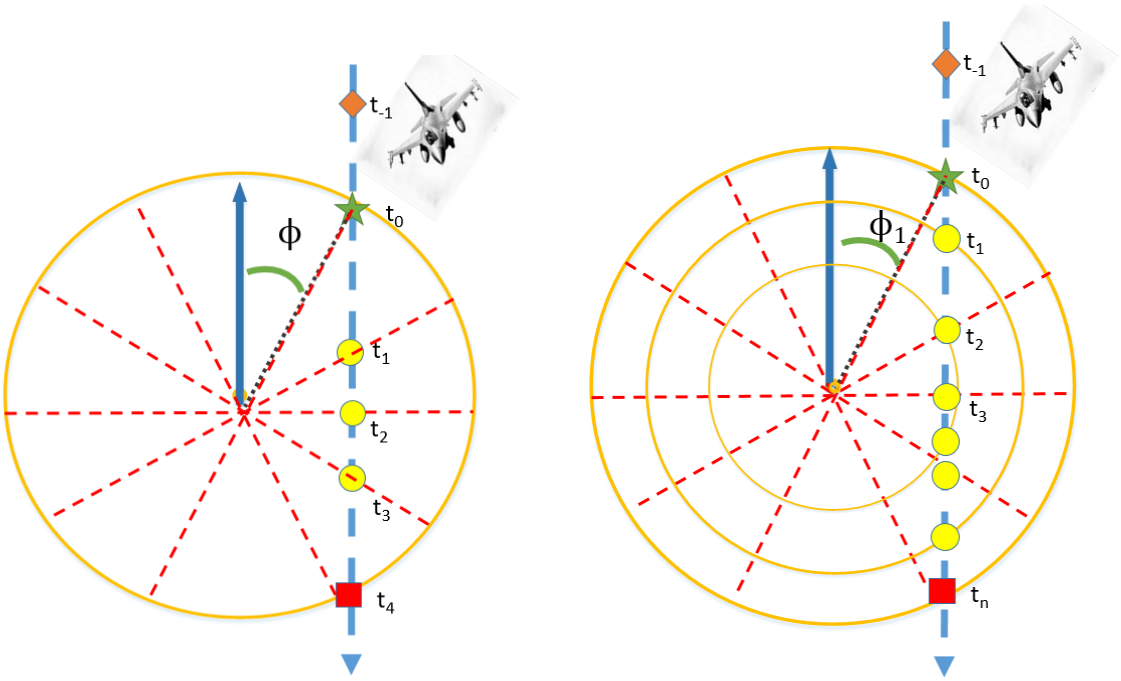
In Figure 3.5a, a sensor with multiple MTTD RVs divides the whole detection region into several fine fan sectors. This structure is extended from the two-tier MTTD constant-rate sensor with higher-level resolution in aspect angle to the target RCS response. With the same scenario of the target movement, the target gets scanned by the sensor with time step $\triangle t$ from simulation time t_0 to t_4. In each scanning event, the P_d of the target may be different due to the different MTTD in the region. This approach considers the aspect angle varying as the target is moving across the sensor. However, the range between the target and the sensor is not considered. A further model experiment is proposed next.

3.2.2 Multiple Block-Sectors MTTDs Sensor

In Figure 3.5b, the detection region of the sensor is divided into multiple block sectors with associated P_d according to the RRE. Each block sector has an MTTD for an RV to represent P_d of the target at the location. The block sectors consider angle factor as the fan sector model dividing the detection region into different angles of sectors. The range factor is represented in different range cells in an angle sector.

3.2.3 Conclusions and Challenges of Multiple-Sectors MTTDs Sensor

Based on the two advanced multiple sectors of MTTDs sensor models, a conclusion can be drawn that the more dividing sectors there are in the sensor detection region, the more detailed resolution of P_d associated to the range and aspect angle between the sensor and the targets. The multiple sectors sensor model seems to be the right approach for upgrading the simple one constant-rate sensor into the advanced sensor model that contains more



(a) A sensor detection rate divided into fan sectors (b) A sensor detection rate divided into block sectors

Figure 3.5. The demonstration figures for illustrating multiple-tier constant-rate sensors. The sensor detection rates are divided into small sectors.

details of range and aspect angle information of the targets can be determined. However, this improved approach has two inevitable challenges:

1. **Model construction complexity:** The complexity of model construction with DES structure depends on the number of possible state transitions in the model. For the multiple-sectors sensor model structure being considered, each sector gets considered as the state transition region, and all states of the model are required to be constructed into mediator and referee entities in order to determine the time and location that targets enter the regions. A correspondingly tremendous increase in the number of states can make it extremely difficult and complex to construct a sensor model with multiple MTTD RVs in multiple sectors.
2. **Model execution complexity:** Assuming the multiple sectors sensor under DES structure is feasible and completed, the model execution efficiency can be low due to the computational latency of each sector response for the target travel through

the sensor. Recall the scenario from the two-tier MTTD sensor model; the event schedule of target pass through the sensor will be checking if the detection happens in the region 1. If the detection does not happen in region 1, then schedule a determining process repeatedly for the region 2 and so on for more regions. By these iteration processes, the DES structure sensor loses the advantage of executing efficiency when the model contains too many possible transition states.

By considering the two major challenges just mentioned, a sensor model that considers the aspect angle and range to the targets is necessary. Thus, a more efficient model structure framework is required and will be proposed in the later sections.

3.3 Model III: Hybrid Sensor Model

The multiple-sectors DES sensor, proposed in the previous section, presents challenges when attempting to integrate parameters that have complex outcomes, such as the complex RCS responses of moving targets. Due to the complexity of building a model with so many fractional detection sectors and the tremendous deterioration of execution efficiency by processing detection events at each sectors along with the target waypoint, the multiple-sectors sensor structure is not an efficient implementation for this approach.

To address such problems, this research categorizes all high-fidelity parameters that have been considered in the model and different simulation time mechanism models into additional two categories:
1) The model integrates an atemporal model, which has a timeless factor but can lead to noticeable computational latency, such as RCS value of the target in this study.
2) TS simulation methodologies are integrated into a DES framework to build a flexible hybrid sensor model. The TS mechanism is utilized to emulate the sweeping behavior of the search radar sensor. The sensor has a scan of the target once in every rpm.

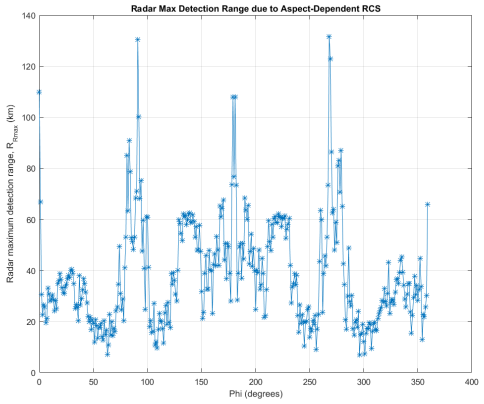
This hybrid structure can integrate the highly detailed factors of atemporal and TS sensor structure with the reasonable computational efficiency of DES models.

3.3.1 Atemporal RCS Simulation Response

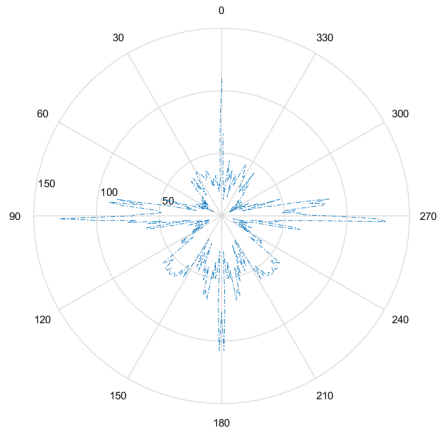
The RCS value of the target to the sensor is the function of the aspect angle regardless of the time factor with other RRE parameters are configured as simulation design, which are listed in Figure 2.20. The RCS values of the target are intensively computed by the high-fidelity, physically based, simulation software CST studio [43]. The pre-computed RCS values table of the target is in 360° orientation along ϕ with $\theta = 90°$ and 0.5° degree resolution. The RCS model of the F-16 Falcon fighter is shown in Figure 2.25.

Due to the noticeable computational latency and the time independent characteristic of the RCS model, this model is classified as an atemporal factor in the simulation construction.The computational latency is isolated in order to execute it in a real time or differential simulation time scenario, such as DES.

In Figure 3.6a, the Rmax is calculated based on the RRE with the RCS value in $0° \leq \phi \leq 360°$ with a 0.5° increment. The Rmax reaches 110 km to 130 km around $\phi = 90°, 180°, 270°$, while the Rmax is down to 15 km to 20 km around $\phi = 60°, 300°$. Figure 3.6b shows the Rmax in a polar plot. The range distributes identically with the RCS model of the object with different scaling value due to the computation by the RRE. The Rmax database of the target is carried by the sensor entity for detecting decision making. The event graph of the sensor model that carries the target Rmax database is shown in Figure 3.12.



(a) The Rmax of F-16 in $\phi = (0°, 360°)$



(b) The Rmax of F-16 polar plot in $\phi = (0°, 360°)$

Figure 3.6. The Rmax of an F-16 Falcon fighter in azimuth angle with $\theta = 90°$. In (a) the Rmax of the object is computed by the RRE with the RCS model, which is pre-calculated by CST studio. In (b), the polar plot of the Rmax of the sensor to detect the target, which is calculated by the range equation with the RCS.

3.3.2 TS Structure

TS structure is applied to emulate the sweeping mechanism of a search radar antenna for each scan. Since the sensor scans at the target once every rpm, a ping event is constantly scheduled in sweeping intervals to represent the antenna sweeping behavior in the Ping Angle Mediator entity. The event graph representation is shown in Figure 3.7. The mediator entity determines the Rmax of the sensor based on the current position and the aspect angle of the targets. The Rmax is calculated by the RRE with the RCS viewed by the sensor at each sweeping event.

3.3.3 Hybrid Sensor Architecture Event Graphs

The hybrid sensor model is a DES-based architecture integrating the atemporal RCS model from an arbitrary object and the TS simulation behavior to emulate the antenna sweeping mechanism. The event graphs of two distinct entities in the hybrid sensor model are introduced.

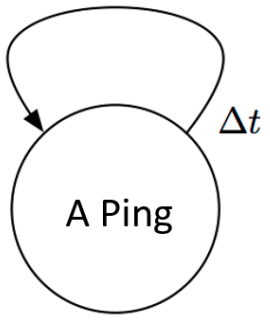


Figure 3.7. A constantly scheduled "A Ping" event is scheduled by a fixed time-step scheduling interval Δt. The TS mechanism emulates the fixed interval antenna rpm.

RCS Sensor Entity Event Graph

Unlike the cookie cutter sensor that contains only "Detection" and "UnDetection" events, the RCS sensor entity, which is shown in Figure 3.12, contains detailed parameters that integrates RRE and atemporal RCS factor into the hybrid architecture.

- **Parameters:** Parameters are the initial values or data that are designated during the simulation initialization state. These variables do not change during the simulation and can be accessed by the local class or globally by other classes, depending on the entity design.
  - **AngleRangeArrayMap:** It is the Rmax database of the target. The Rmax table contains the detection range of the target to the sensor with the aspect angle accordingly. During the detection process, the Rmax will be referred according to the current aspect angle of the target.
  - **pinIteger:** This is the parameter of the rpm of the antenna.
  - **isSwerling:** A Boolean parameter to control whether the RCS fluctuation loss is enabled.
  - **RCSSwerlingFactor:** The RV represents the fluctuation loss of the target. A $\chi^2_{df=4}$ RV is applied in this study for representing the model of Swerling case III.
  - **thetaAZ:** The parameter of antenna 3 dB effective beamwidth.
  - **prf:** The radar pulse repetitive frequency.
- **State variables:** The state variables are computed explicitly during event state transition according to the parameters inside the entity class or from interactively accessing other entity classes.

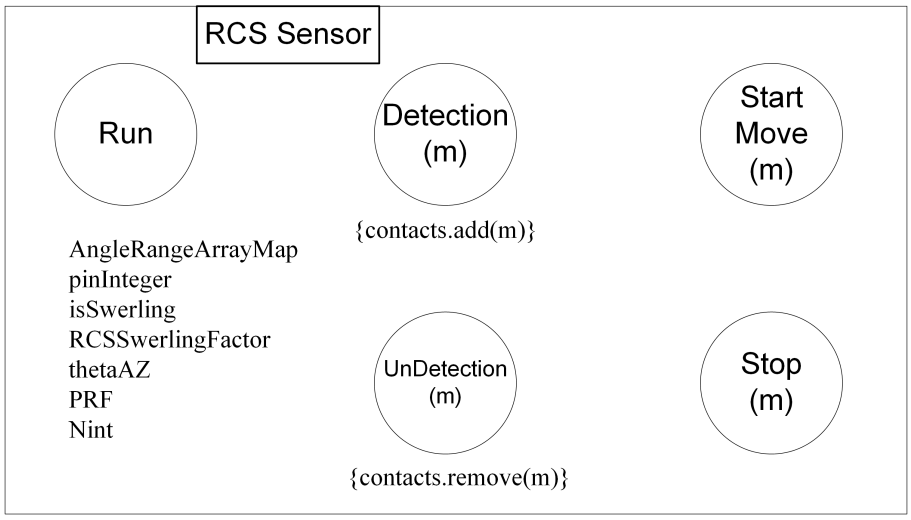


Figure 3.8. RCS sensor event graph for DES detection of independent target entities.

  - **Nint:** The parameter of pulse integration improvement factor. The computation is based on the two parameters "theatAZ" and "pinIteger." The detailed computation has been introduced in section 2.3.1.

Ping Angle Mediator Entity Event Graph

This mediator entity determines whether the "Detection" event happens by checking the position and aspect angle between the target and the sensor once every rpm. The parameters and the state variables are listed as follows:

- **Parameters:**
  - t_{1st}: It is the time delay for scheduling the first "A Ping" event when "EnterRange" event is scheduled by the "Referee" entity class. $t_{1st} \sim uniform(0, \Delta t)$, which is a uniform distribution RV bonded between 0 and the radar antenna rpm.
  - Δt: It is a fixed time interval for the self-scheduling "A Ping" event. Δt = rpm.
- **State Variables:**
  - **Angle:** Finding the current aspect angle between these two entities.
  - **RangeMax:** Based on the current aspect angle, the Rmax of the target to the sensor is read out from the sensor Rmax database.

- **MSDistance:** The current distance between the sensor and the target are calculated.

The decision making sequences are listed as follows:

1. The Ru represents the maximum detection range of the sensor for "Referee" to determine the time of the target making contact with the sensor. When the contact occurs, the "EnterRange" event is announced by the "Referee" entity class. The "Referee" isolates the hybrid sensor architecture from the irrelevant and insignificant events outside the Ru for boosting simulation executing efficiency.
2. Once the "EnterRange" event is scheduled by the "Referee" entity class, the first "A Ping" event is scheduled with a time delay $t_{1st} \sim uniform(0, \Delta t)$, which is an uniform distribution RV bonded between 0 and the radar antenna rpm. Figure 3.9 demonstrates the three possible antenna revolution angles ρ_1, ρ_2, ρ_3 out of 360° when the contact of the target occurs. For a radar antenna sweeping in a constant angular velocity $\omega = rpm = \Delta t$, the antenna revolution angle ρ when a target contact occurs is uniformly distributed between 0° and 360°. The probability of each antenna revolution angle ρ_n when the target contact occurs is the same. Therefore, the minimum time delay for the first effective scan from the antenna to the target is 0 when the contact occurs at the antenna revolution angle $\rho = 0°$. The maximum time delay for the first scan happening is to wait for a whole antenna revolution cycle, which is the parameter "rpm" in the model. It is when the contact occurs at the time when the effective scan angle has just passed and happening at $\rho = 1°$ for example.

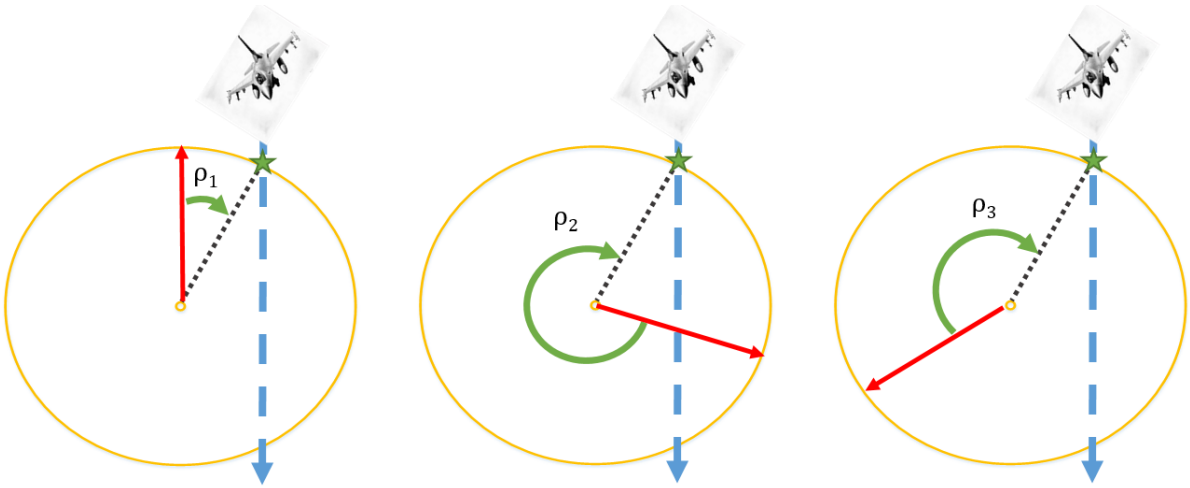


Figure 3.9. Hybrid mediator event graph ensures fair adjudication of whether detection occurs.

3. When the the "A Ping" event happens, the mediator checks the current geometry location between the target and the sensor and computing the distance and the aspect angle based on the current location of these two entities. Three state variables are calculated in this event:
   - Angle
   - RangeMax
   - MSDistance

   The "Detect" event does not necessarily happen at each "A Ping" event due to the detecting decision condition in Equation 2.5.
4. If the current Rmax is larger than the current target distance, this means the target is within the detection range of the sensor. The "Detect" event is announced and scheduled by the mediator entity at the same time as this "A Ping" event.
5. If the current Rmax is smaller than the current target distance, the target is not seen by the sensor. Another "A Ping" event will be scheduled after one antenna rpm = Δt and then checking for the "Detect" event as the logic in step "3." In Figure 3.7, the "A Ping" event is scheduled repeatedly in a fixed TS interval: Δt = rpm before the "Detect" event condition is satisfied. The chance for the sensor to update the information of the targets happens only once every rpm.
6. When the "ExitRange" event is scheduled by the "Referee" entity, the "UnDetect" event will be scheduled in the same time and the next "A Ping" and "Detect" events

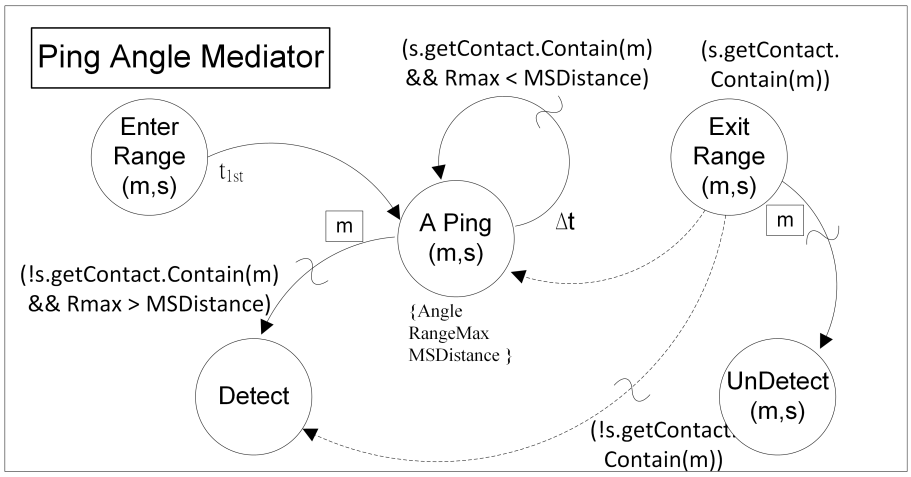


Figure 3.10. Hybrid mediator event graph ensures fair adjudication of whether detection occurs.

will be canceled to prevent conflicting schedules in the future simulation time.

Whole System Assembly Event Graph

Because the whole hybrid sensor model scenario is based on the DES architecture, the subclass entities have interaction with each other based on events with the same name and signature along with the direction of the event listener pattern. For each entity component in this model, it is not necessary to have the "Run" event activated in the simulation. With the connection of the event listener pattern between entities, the events of listener entity components are scheduled by the identical events in the source entity components.

The blue dashed box in Figure 3.11 shows the event transiting relationship between "Path Manager," "Simple Mover," and "RCS Sensor" entities. The corresponding events in the entities, which have the same name and signature, are marked in the same color for clear readability.

The simulation event scheduling algorithm is as follows:

1. The simulation starts at the "Run" event (light pink circle) in the "Path Manager" entity, while other entities do not have a scheduling edge from "Run" event for the event scheduling.
2. The "Start" event (white circle) is scheduled for starting the simulation. The "MoveTo"

event (blue circle) is scheduled in the same time with the first waypoint for the mover.

3. The same event in "Simple Mover" is scheduled in the "Simple Mover" because the event listener hears the event from the "Path Manager" to the "Simple Mover."
4. When the "EndMove" event (orange circle) in the "Simple Mover" happens, the same event is triggered and is scheduled at "Path Manager."
5. The "Path Manager" also listens to the "Stop" event (red circle) from the "RCS Sensor" and "Simple Mover" for determining if the next waypoint is assigned to the "Simple Mover" and "RCS Sensor" entities.

Since the "RCS Sensor" is basically stationary at one location in this scenario, there is no waypoint assigned to the sensor entity.

The yellow dash-dotted box in Figure 3.11 manages the sensor detection algorithm by the "Referee" and "PingAngleMediator." The detecting decision making sequence is as follows:

1. "Referee" determines when the "EnterRange" (light blue circle) is scheduled by listening for the "StartMove" event (green circle) and gets the waypoints from both "Simple Mover" and "RCS Sensor" entities. The time when the target gets contact to the maximum detectable range of the sensor is calculated by "Math2D. findIntersectionTimes(m, s)" in Simkit based on the waypoints.
2. When the "EnterRange" event happens in "Referee," the same event is scheduled in "PingAngleMediator" and then the "A Ping" event (white circle) starts being scheduled and checking if the target enters the Rmax repeatedly in the interval of antenna rpm. The target aspect angle and the Rmax associated with this angle are constantly updated in each "A Ping" event.
3. Once the "Detect" event (cyan circle) is announced by the "PingAngleMediator," the same event is listened for and registered, and the target is detected by the "RCS Sensor" entity.
4. The "ExitRange" event (purple circle) is calculated and scheduled by the "Referee" based on the waypoints of "Simple Mover" and "RCS Sensor." Once the "ExitRange" event happens in "Referee," the same event will be triggered in "PingAngleMediator" and then the "UnDetect" event (gray circle) gets scheduled in the same time.
5. The "UnDetect" event announced by the "PingAngleMediator" will be heard by the "RCS Sensor," and the same event will be scheduled for the unregistered detection of

the target.

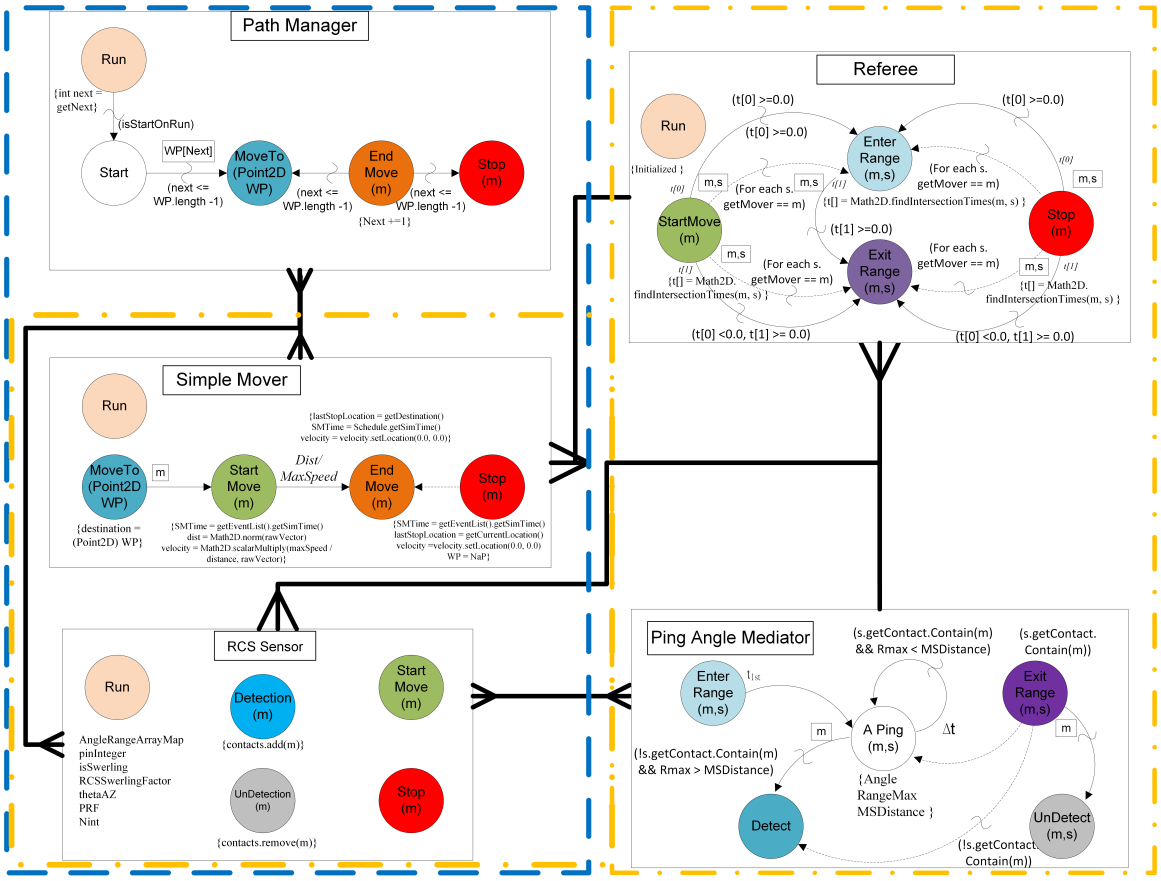


Figure 3.11. Hybrid sensor model overall DES event graph.

3.3.4 Simulation Scenario

Figure 3.12 demonstrates a scenario of the detection decision processes of the hybrid sensor model architecture as an aircraft closes in on the sensor. The simulation starts at t_{-1} as the aircraft is outside the Ru at the orange diamond. The detecting decision algorithm is computationally demanding and is not be scheduled when the target is outside of the Ru. This DES structure saves computational cost from irrelevant events outside the possible detectable range of the sensor. Recall the detection-decision algorithm in Equation 2.5; the target detectable distance R_{detect} has to be shorter than both Ru, which is the yellow rim, and Rmax, which is the blue line figure in the center. The blue line figure is the target detection range as viewed by the radar in a different aspect angle that depends on the RCS.

Table 3.4. Simulation event schedule for hybrid sensor scenario

Simulation time	Scheduled event
t_{-1}	Target approach , not yet detectable
t_0	Target enters detectable range
$t_1 = t_0 + t_{1st}$	A radar sweep
$t_2 = t_1 + \triangle t$	A radar sweep
t_2	Target is detected
t_3	Target exits range
t_3	Target is undetected

The first sweeping event is scheduled after the target reaches the sensor Ru (green star on perimeter) with the time delay of an uniform RV $t_{1st} \sim (0, \triangle t)$. The sensor starts repeatedly scanning to the target in a fixed TS with a period of $\triangle t = t_2 - t_1 \cdots = t_n - t_{n-1}$ while the target moves in distance $\triangle d = s \cdot \triangle t$ at each TS. At each sweeping time step t_1 and t_2, the aspect angle of the target changes from ϕ_1 and ϕ_2 respectively. Precise RCS is available corresponding to angle changes for Rmax, which are computed from Equation 2.3. The target does not reach the Rmax even though it is within Ru at t_1 (the first yellow circle).

When the target arrives at t_2 (the second yellow circle), the target has traveled the distance $= t_{1st} \cdot s + \triangle d$ since entering the Ru. The target's position is within Rmax (the blue line figure in the center). That is the moment the sensor declares the target is detected. The full simulation event list is shown in Table 3.4.

The sensor might never detect a target if the target never reaches Rmax during all time steps. The parameters attributed to the sensor model are:
1) Target RCS table, 2) Target speed, 3) Target aspect angle, 4) Radar sweeping period, 5) Radar prf, 6) Antenna 3dB azimuth beam-width.

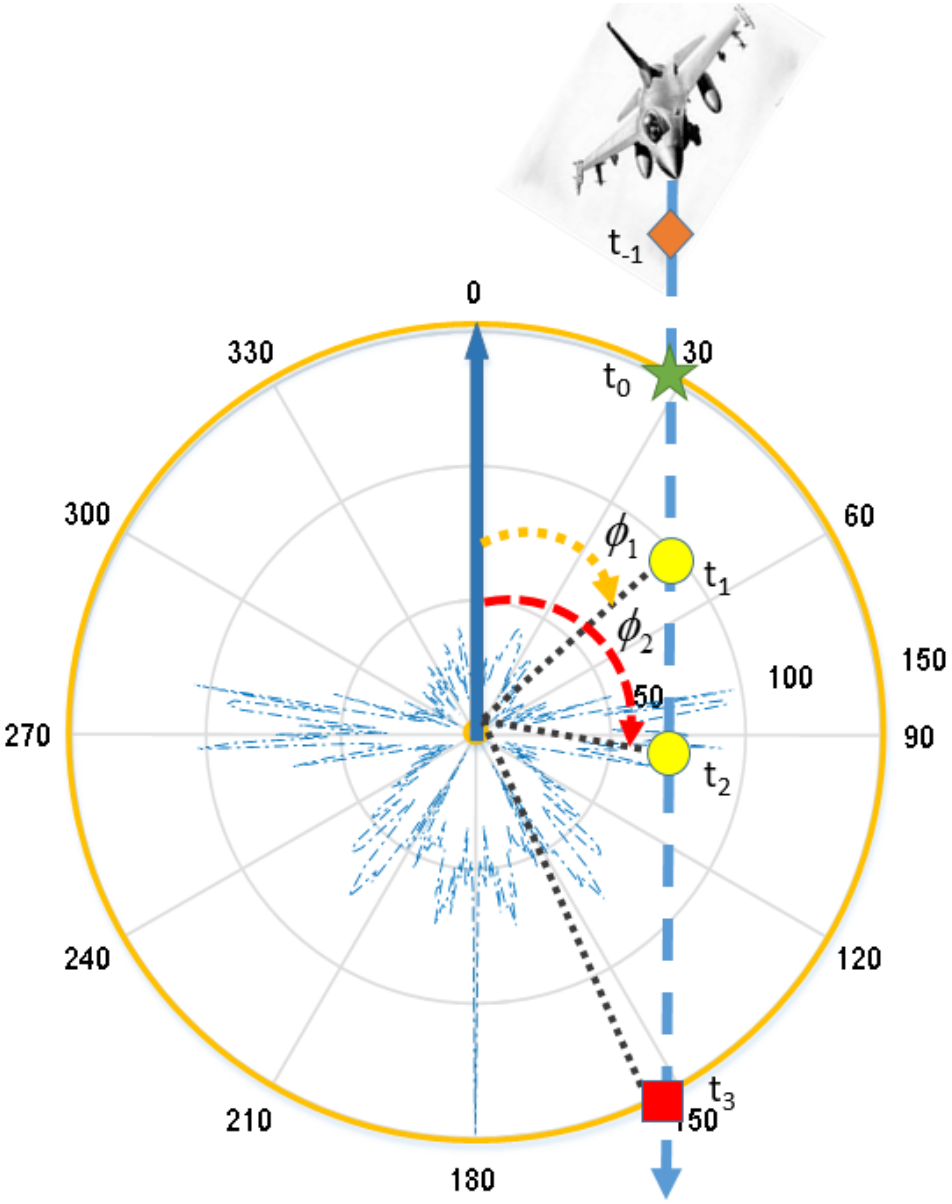


Figure 3.12. The hybrid sensor operation scenario. The blue figure represents the aircraft detection range as viewed by the radar depending on radar cross section (RCS)

3.3.5 Simulation Sequence Diagrams

In this section, a Unified Modeling Language (UML) sequence diagram is introduced for illustrating the event transition between the five entities and the time schedule in the simulation. The UML is the successor to the wave of object-oriented analysis and design

methods. The UML went through a standardization process with the Object Management Group (OMG) and is now an OMG standard. The UML is called a modeling language, not a method. The modeling language is the important part of the method. It is the key part for the simulation practitioners to communicate the design of the model.

With the same simulation scenario, Figure 3.13 is a UML sequence diagram of the hybrid sensor model in this simulation scenario. The vertical line is called the object's lifeline. The lifeline represents the object's life during the interaction [51].

The five entities in the hybrid sensor model are listed in the sequential diagram from left to right: RCS Sensor, Ping Angle Mediator, Referee, Simple Mover, and Path Manager. Each message is represented by an arrow between the lifelines of two objects. The order in which these messages occurs is shown top to bottom on the page. The "Simulation Time" block is the event scheduling time sequence, which matches to the simulation event schedule in Table 3.4. The events announced by the entities are in the rectangular boxes under each of the entities for indicating the source of the events.

The blue-boxed events are in the advanced time structure, also known as DES in the study. The "Path Manager" announces "Move To Waypoint" events to the "Simple Mover" entity when the "End Move" event is fired by the mover. The "Simple Move" schedules the "End Move" event when it receives the waypoint from the path manager with the "Start Move" event. The duration of the start move event $t_4 - t_{-1}$ is calculated by the distance of the waypoint and the speed of the target. The "Enter Range" and "Exit Range" events are announced by the "Referee" with the duration $t_3 - t_0$, which is calculated by the range of Ru in the current waypoint and the speed of the target.

The yellow-boxed events are scheduled in a fixed TS. The purple oval box represents the atemporal RCS factor in this research. This entity provides the updated RCS value for the "Ping Angle Mediator" at each TS antenna sweeping event based on the current aspect angle of the target.

The blue-diamond box represents the condition for the following event. In this scenario, when "if detection" event is false, "Perform a radar sweep" event will be scheduled repeatedly in an interval of $\triangle t$.

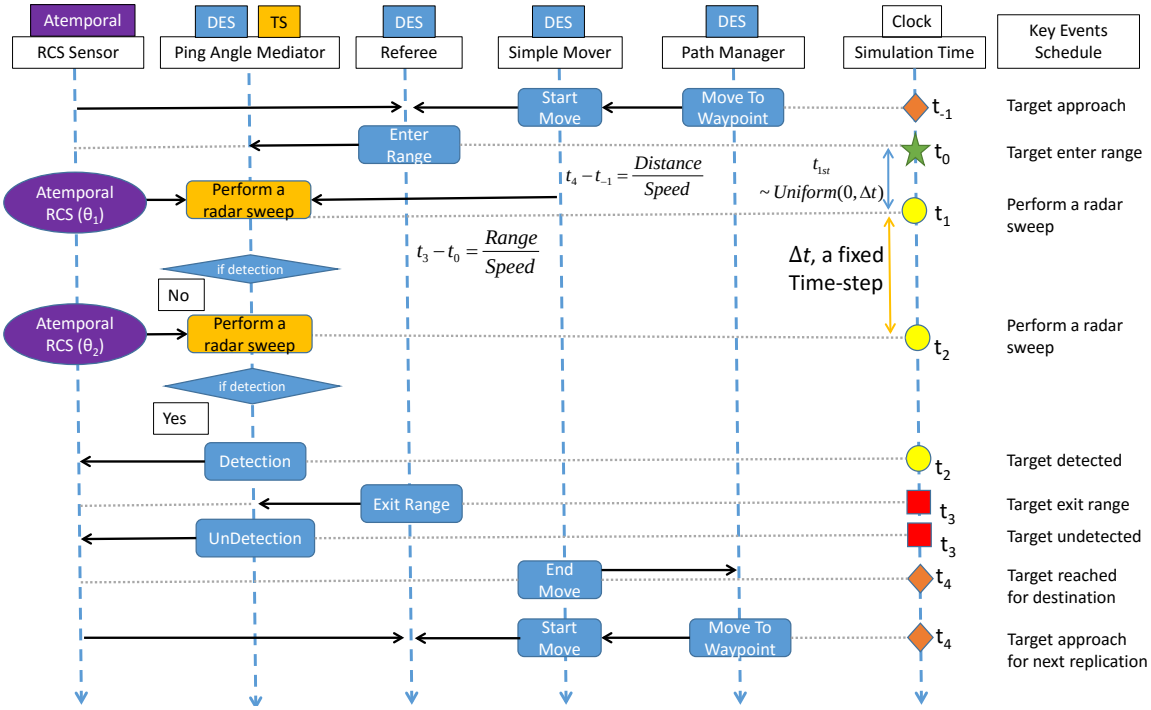


Figure 3.13. UML time sequence diagram showing event-response progression between DES, TS, and Atemporal simulation entities of a hybrid sensor simulation scenario.

The "Detection" event is fired by the mediator when the detection condition is "true". The "Detection" event is received by the "RCS Sensor" for registering the detected targets.

3.3.6 Conclusions

The hybrid sensor construction framework presented here successfully integrates three different simulation-timing structures and also introduces high-detailed radar parameters into the model-detection algorithm. The hybrid sensor construction has greatly mitigated the incoherent timing issues between each unique timing mechanism simulation structure, and the hybrid high-fidelity model can be executed efficiently and accurately as the typical characteristics of DES. The overall scenario construction is built by multiple functionally independent classes with mutual event interactions occurring in between.

Based on this flexible hybrid-sensor architecture, the RCS varying target-sensor models are extended to eight more aircraft and missile models, which can be applied into multiple

targets dynamic simulation scenario in future work. The RCS-varying available models in this research are:

- **Conventional Military Aircraft:** The traditional military aircraft represent the middle size RCS aircraft models.
  - F-16 Falcon Fighter is the example aircraft model that has been used for the prototype hybrid sensor model in this research.
  - F-18 Hornet Fighter
- **Commercial Aircraft:** The commercial aircraft represents the large size RCS aircraft models that have longer Rmax on average to the sensor.
  - B-737 400
  - B-747 400
- **Stealth Aircraft:** The Stealth aircraft has a relatively small RCS in comparison with the conventional aircraft with similar physical size. These models have shorter average Rmax than the similar size aircraft to the sensor.
  - F-35 Lightning Fighter
  - B-2 Spirit Bomber
- **Unmanned Aerial Vehicle (UAV):** The UAVs are a new type of tactical aircraft with a size that can vary from that of a regular aircraft down to miniature size.
  - MQ-1 Predator UAV
- **Missiles:** The missile objects have small RCS in the front size and relatively larger RCS in the broadside.
  - AA-11 Archer SAM

The detection response of each aircraft entity is based on its own RCS viewed by the sensor entity by the specific aspect angle. The aircraft models are obtained from open commercial and unclassified resources. The RCS are computed in full-size model with the assumption of PEC material. The model RCS results may not be ideally matched to the real object due to the imperfection of the model geometry and the lack of detailed covering material information. Nevertheless, the RCS results still provide detailed high-resolution information based on aspect angle, which affects the sensor detection ability to the targets. All available models' RCS data and the maximum detectable range to the sensor are shown in Appendix B.

3.3.7 Challenges

There are some intrinsic characteristics of hybrid sensor model structure that still make the model structure complicated and cumbersome in a certain sense.

- **Model Execution Efficiency:**
  The TS structure, which is used to emulate the radar antenna sweeping mechanism periodically, can raise the number of scheduling events of target detection. The frequent antenna sweeping events degrade the efficiency of the sensor model under DES structure. The more trivial events that have been scheduled, the less efficient the DES structure is and the closer to TS structure of the model.
- **Model Structure Complexity:**
  In order to incorporate the crucial dynamic angle and range factors into the sensor-target simulation model, the model constructing complexity of the hybrid sensor model is much more efficient and less difficult than the immature proposed fractional MTTD region sensor structure in section 3.2. However, the sensor model carries the pre-computed atemporal RCS response table of the target. The RCS table makes the sensor model append more data into the structure.

To mitigate these inherent drawbacks of the hybrid sensor framework, regression analysis is next applied for forming simplified structure models with equally likely performance, and the comprehensive and efficient DoE are required for analyzing the response and behavior of the hybrid sensor model. The further simplified sensor-model response approximations are introduced in section 4.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Design of Experiment (DoE) for Hybrid-Sensor Models

There are three principal means of acquiring knowledge: observation of nature, reflection, and experimentation. Observation collects facts; reflection combines them; experimentation verifies the result of that combination.

- Denis Diderot [52]

This chapter introduces an approach to analyze the hybrid-sensor model behavior by executing multiple simulations with proper DoE designs. In experimental design terminology, the input parameters and structural assumption composing a model are called "factors," output performance measures are called "responses" or "observations." The decision of parameters and structural assumptions are considered fixed a spects o f a model in a range of the goals of the study. These factors are usually in several different responses or performance measures of interest [53].

Thus the DoE in this study needs to provide the comprehensive insight of the hybrid-sensor model behavior in the range of values for the parameters of interest. The observations generated from the DoE are applied to regression analysis for studying the significant factors of the model for further forming a meta-model that retains the equally likely behavior as the hybrid-sensor model with much less structure complexity and more computational efficiency. There are two primary considerations for a proper DoE:

- For having a deeper insight into the comprehensive behavior of the models, the number of available experiments must be statistically significant for analyzers. Nevertheless, because of realistic constraints such as budget, time, or available exercise space to conduct experiments, experimenters are usually unable to get unlimited experimental results for analysis. Efficiently designing experiments is important.
- A well space-filling measurement of the system parameters from the DoE is important. The more detailed levels are included in the DoE, the more the relationship between system and parameters will be captured by the analysis. This need for detail must be balanced with the need for efficiency. A well-covered and succinct DoE is necessary.

4.1 DoE by Full Factorial Design

To analyze the interactive relationship between all input parameters in the model, a DoE that covers space in the range of parameters and the orthogonality between factors is crucial to reveal the relationship between parameters and system state. If a model has only one factor, the DoE is conceptually simple: the simulation is run at various values or levels of that factor. However, now suppose there are $k(k \geq 2)$ factors and the effect of each factor's value are studied as well as the factor's interaction with one another, i.e., whether the effect of one factor on the response depends on the levels of the others.

A 2^k factorial design chooses two levels for each factor and the calls for simulation runs at each of 2^k possible factor level combinations. These combinations are called design points (DPs). Because there only two levels for each factor, the response is assumed to be linear or monotonic over the range of the factor. If the response is non-monotonic (e.g., in the shape of a parabola) over the analyzed range, a misleading outcome can result because that the nonlinear factors have only a linear effect on the response [54].

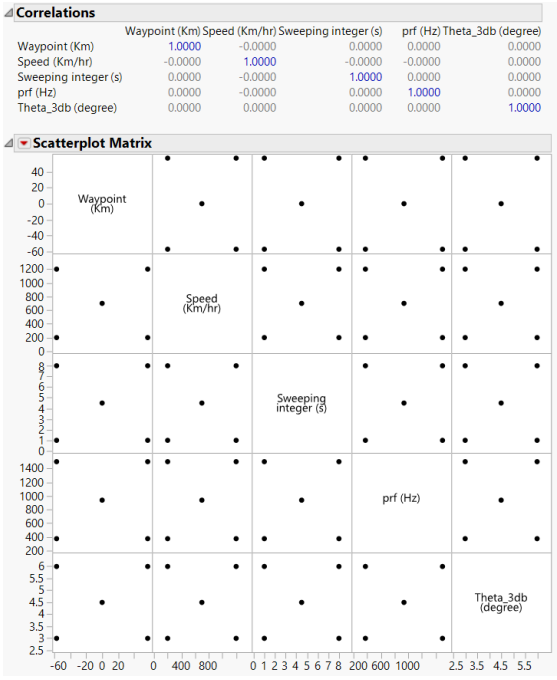
Figure 4.1a shows a 2^5 factorial DoE with five factors in the variable range of the hybrid-sensor model simulation shown in Table 4.1. Each factor has two levels of resolution. The 2^5 factorial design shows the prefect uncorrelated (zero correlation) behavior between each parameter at the top of the correlation table between each parameter. There are $2^5 = 32$ DPs in these five factors, two-level resolution DoE. However, compared to the NOLH 32 DPs DoE in Figure 4.1b, the space-filling in the parameters of 2^5 factorial design is scarce and much more infrequent than the NOLH DoE for a similar number of DPs.

When the parameters have one level increment in the three-level resolution, the 3^k factorial design, the correlation is still zero between each parameter and the number of DoE has an incremental from $2^5 = 32$ to $3^5 = 243$. Nevertheless, compared to nearly the same of amount of 257 DPs NOLH DoE in Figure 4.1d, the space-filling in the parameters range is still quite scarcely spreading out in 3^5 factorial DPs, as shown in Figure 4.1c. This sparse 3^5 factorial DoE will not capture enough detail information on a complex behavior model such as the hybrid-sensor model in this study.

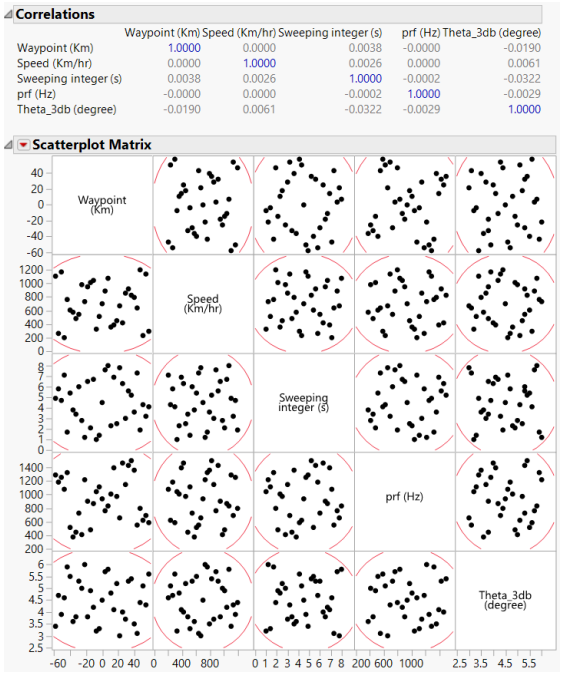
The parameters space-filling efficiency in full factorial design is much less efficient than in NOLH DoE. When the space filling of parameters is sparse, the DoEs do not capture detailed

levels of behavior for the model during regression analysis. The DoE skips many possible values of parameters and loses a large portion of the observations of model information at the uncovered region of parameters in the DoE.

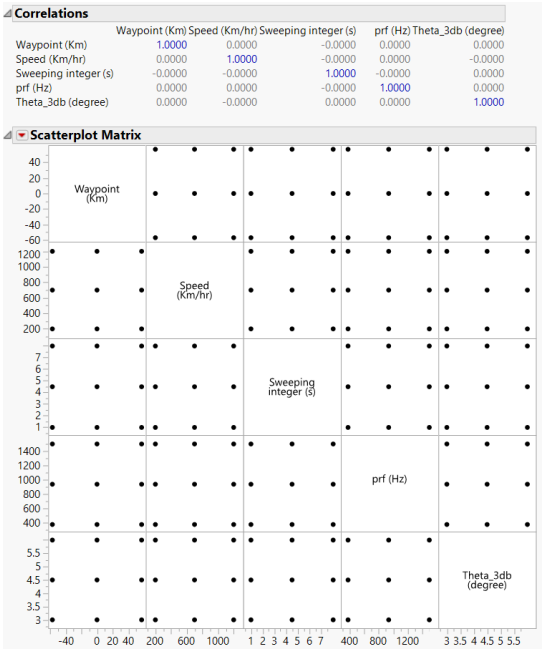
The goal of Space-Filling Design (SFD) is to spread the DPs "uniformly" throughout the experimental region. The efficiency of SFD is particularly important for the factors to be continuous variables or discrete variables with potentially a large number of different levels [55].



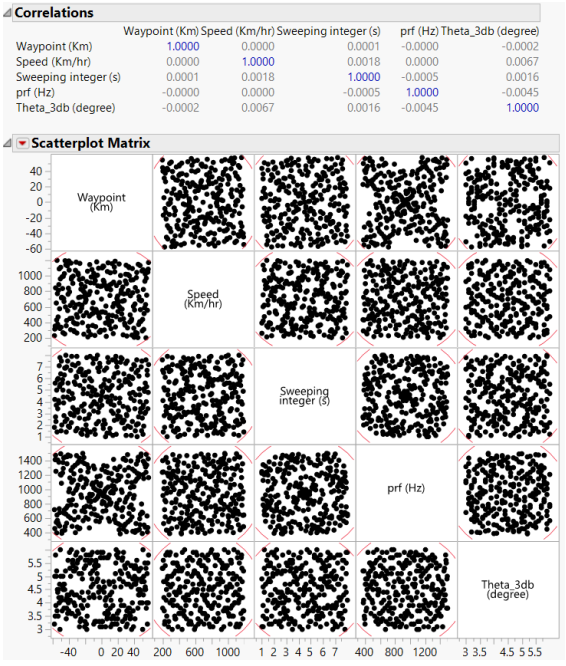
(a) 2^5 factorial: 32 DPs with parameters in 2 levels



(b) NOLH: 33 DPs with 5 parameters



(c) 3^5 factorial:243 DPs with parameters in 3 levels



(d) NOLH: 257 DPs with 5 parameters

Figure 4.1. Scatterplot matrices of selected factorial and NOLH DPs. Adapted from [56].

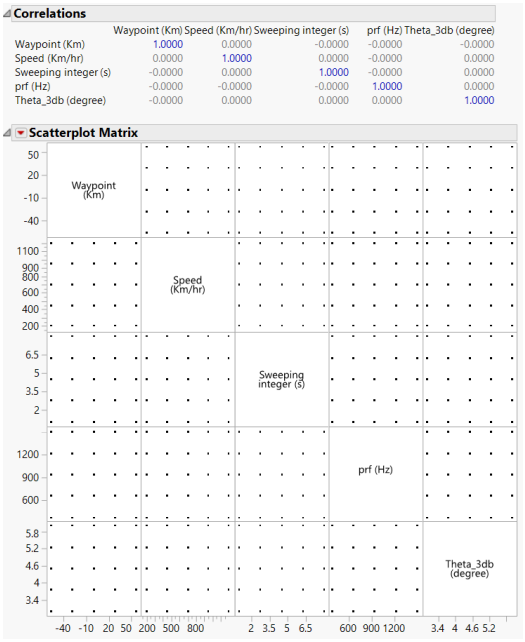
4.2 DoE by NOLH Design

When the five parameters resolution is increased to five levels in full factorial design, the number of DoE increases to $5^5 = 3125$, as shown in Figure 4.2a. The DPs keep the perfect non-correlation characteristic between parameters, but the space-filling efficiency is still insufficient for the highly detailed model analysis.

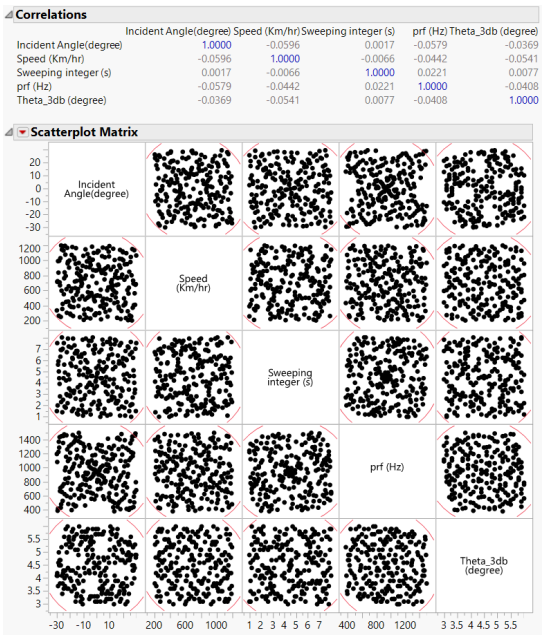
In full factorial design, if applying five factors with 250 levels of each factor, the number of DPs = $250^5 \simeq 9.7 \times 10^{11}$. For each DP executing 100 repetitions, the total simulation execution amount is up to $\simeq 9.7 \times 10^{13}$. Even though in this computer based simulation the number of experiments that can be conducted is not restricted, this execution amount is still computational overwhelming.

The NOLH presented by Cioppa et al. [18] provides an efficient space-filling DoE that allows a view of 257 levels, varying of up to 29 factors in 257 design points. With 100 repetitions, the total number of simulations to process is 25, 500. This makes a 3 billion-to-one difference on simulation execution tractability. Figure 4.2b shows that the design points of each factor fills in the DoE space quite evenly, with only minor overlapping correlation to each other. Compared to the full factorial DPs in Figure 4.2a, the five factors parameter with resolution to five levels, the full factorial DoE has 3125 DPs while the NOLH DoE only generates 256 DPs. Furthermore, NOLH DoE covers much more parameter range space than the full factorial DoE. The NOLH does not provide perfectly orthogonal DoE between each factor like full factorial design. However, the concise DoEs generated by NOLH are a crucial solution for executing comprehensive interaction simulations of high-complexity models with many factors, and many levels in each factor, sacrificing only a minor correlation between factors. This concise and highly efficient space-filling DoE makes a simulation with high-level parameter resolution feasible, which is not otherwise executable in the full-factorial DoE.

In the simulation scenario, the hybrid-sensor model is located at the origin coordinate point $S(0, 0)$ with the maximum detection range = 50 km (green line). The target initial position is at 100 km north from the sensor at $a(0, 100)$ (red square), which is displayed in Figure 4.3a. The waypoints are designated by the DoE.



(a) 5^5 factorial: 3152 DPs with parameters in 5 levels



(b) NOLH: 257 DPs with angle transformation

Figure 4.2. Five factors DoEs for hybrid model simulation in full factorial vs. NOLH design.

The DoE is set up by NOLH with five parameters in 257 DPs. All parameters are in the range that are listed in Table 4.1 and explained as follows:

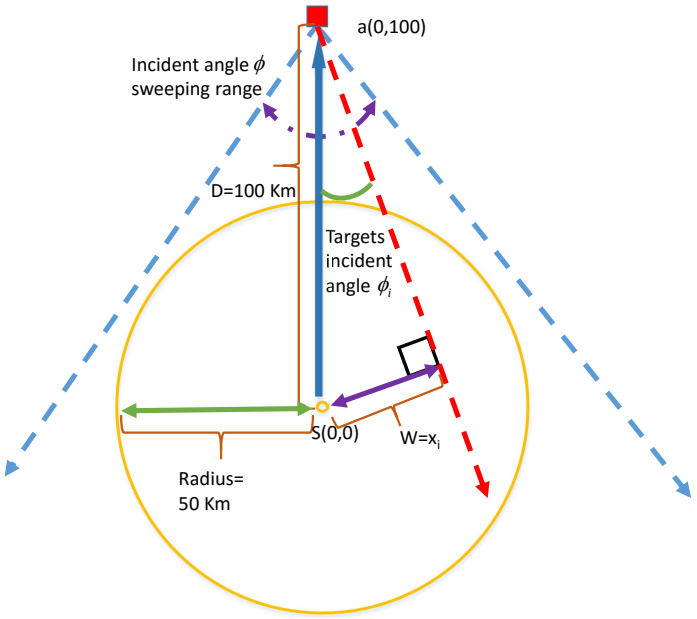
- Waypoint(km): Target destination waypoint x-axis components.
  For simpler waypoint arrangement in the simulation, the waypoints are assigned in a vertical path (red-dotted arrow) and sweep horizontally along the x-axis (purple-dotted arrow) shown in Figure 4.3b. The right bound of waypoint starts from $a_R(57.7, 100)$ down to $b_R(57.7, -100)$ (right blue-dotted arrow), while the left bound of waypoint starts from $a_L(-57.7, 100)$ down to $b_R(-57.7, -100)$ (left blue-dotted arrow). Other waypoints are vertically interpolated in between and travel from $a_i(x_i, 100)$ down to $b_R(xi, -100)$ (red-dotted arrow). The parameter "Waypoint(km)" assigns the different waypoints path along the x-axis x_i that is arranged by NOLH design in the range between -57.7 km and 57.7 km with two decimals level.
  - Incident Angle (degree): Target incident angle conversion.
    Converts vertical waypoints into relative target incident angle to the sensor at a fixed initial position $a(0, 100)$ in Figure 4.3a. The incident angle

presentation provides a clear sense of target waypoints in different incident angles relative to the sensor while the vertical waypoints are convenient for program implementation in the DoE. The target incident angle of sensor is computed by Equation 4.1 as follows:

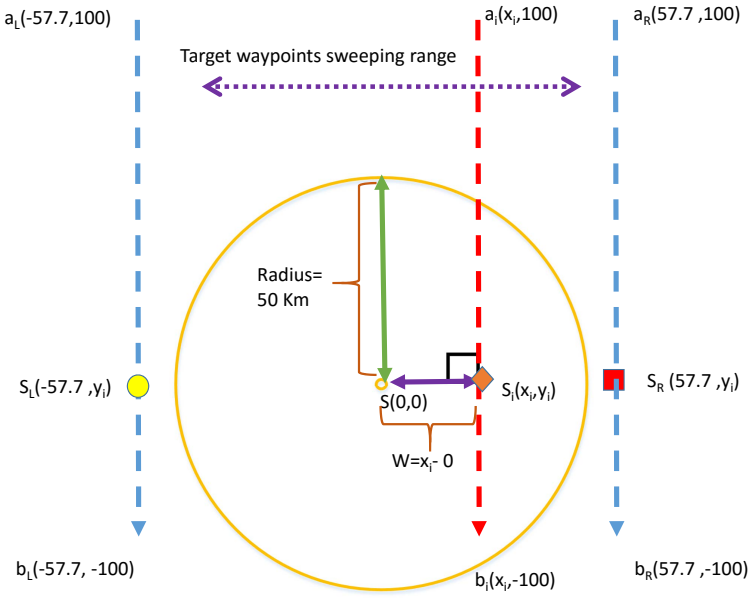
$$Incident\ \ angle\ \ \phi = sin^{-1}\left(\frac{W}{D}\right)\Big|_{W=x_i-0} = sin^{-1}\left(\frac{x_i}{D}\right) \tag{4.1}$$

$$W = ||\overline{SS_i}|| = \sqrt{(0-x_i)^2 + (0-y_i)^2}\Big|_{y_i=0} = x_i \tag{4.2}$$

where D is the radius of the sensor maximum detectable range (green arrow) in Figure 4.3. W is the shortest distance between waypoints vector and the sensor origin (purple arrow) in Figure 4.3b. It is computed as Equation 4.2, because $\overline{a_ib_i} \perp \overline{SS_i}$. When the W are the same in Figure 4.3, the waypoint vector $\overline{a_ib_i}$ in Figure 4.3b is equivalent to the waypoint with incident angle θ_i (red-dotted line) in Figure 4.3a.



(a) The sensor model DoE sweeping incident angle.



(b) The sensor model DoE sweeping along the x-axis.

Figure 4.3. DoE for hybrid-sensor model simulation scenario in incident angle and position sweeping.

- Speed (km/hr): The target flying speed.
  The target is assumed moving in a constant speed along a linear path during the simulation. The speed is in the range from 200 km/hr to 1200 km/hr with two decimals level of precision.
- Sweeping Interval(s): The radar sensor antenna-sweeping period. The hybrid-sensor model in this study emulates the detection behavior of a search radar; therefore, the sensor engages targets once every antenna sweeping period. The antenna sweeping period is designated in a range from 1 second to 8 seconds with one decimal level of precision.
- prf(Hz): The prf. The prf of the sensor is set in the range from 375 Hz to 1500 Hz with one decimal level.
- θ_{3dB}(degree): The radar sensor antenna effective azimuth 3 dB beamwidth.
  Since it is a 2D planar sensor scenario in this study, only azimuth 3dB beamwidth is considered in the sensor parameters. The 3dB beamwidth is in the range from 3° to 6° with one decimal level.

Even though the parameters in Table 4.1 are set up using conventional units of radar applications, all the parameter units have to be normalized identically in the simulation to avoid misrepresenting the quantity of parameters. In the simulation, the unit of the antenna "Sweeping Interval" parameter is converted from seconds to hours in order to match the length unit in the "Target Speed" parameter.

Table 4.1. Factor levels for range and decimals setting by NOLH DoE

Factor name	Way point (km)	Speed (km/hr)	Sweeping interval(s)	prf(Hz)	θ_{3dB} (degree)
Low level	-57.7	200	1	375	3
High level	57.7	1200	8	1500	6
Decimals	2	2	1	1	1

4.3 Hybrid Model Simulation Implementation

This hybrid-sensor model simulation software is built on a Java platform with Simkit library. It is an open source library for DES created by Buss [57]. Figure 4.4 demonstrates two waypoints scenarios of a target approaching the sensor:

1. Scenario 1, the blue-dotted waypoint:
   (a) Simulation time:156.368, the target enters the maximum unambiguous range Ru (orange star) of the sensor with the event called “EnterRange.”
   (b) Simulation time:156.368 to 165.368, starts a series of antenna-sweeping events in the period of “Sweeping interval,” which is designated by the DoE. The sweeping event is called “APinging” in the simulation.
   (c) Simulation time:166.368, the target is detected by the sensor at the yellow circle with the event called “Detection.”
   (d) Simulation time:176.368, the target leaves the range and is undetected by the sensor at the red square with the events called “ExitRange” and “Undetection,” respectively.
2. Scenario 2, the red-dotted waypoint:
   (a) Simulation time:210.04, the target enters the Ru at the orange star called by the "EnterRange" event.
   (b) Simulation time:210.04 to 221.04, the sensor starts a series of antenna sweeping "APinging" events for the detection attempts of the target.
   (c) Simulation time:221.906, the "ExitRange" event occurs, and the target exits the Ru (red square) without any detection event happening since the "EnterRange" event simulation time.

The 257 DoE have been applied on the hybrid model and each DoE run 100 replications. The waypoint DoE is in the range from 57.7 km to -57.7 km while the maximum detection range of sensor model is set up in 50 km radius. This is because the waypoint bounding range is larger than the sensor maximum detection range. The difference is expressed as follows: waypoint range - the sensor maximum detection range diameter = $\langle 57.7-(-57.7)\rangle - 2*50 =$ 15.4 km. This range difference results in 36 out of 257 DPs have been dropped in the simulation since the events that are outside of the maximum detectable will not be processed due to the DES-based construction of the hybrid-sensor model. Therefore, 221 out of 257 DPs contain valid waypoints in the DoE, and the total number of valid data from simulation replications is 22,100 after 100 replications of each valid DP.

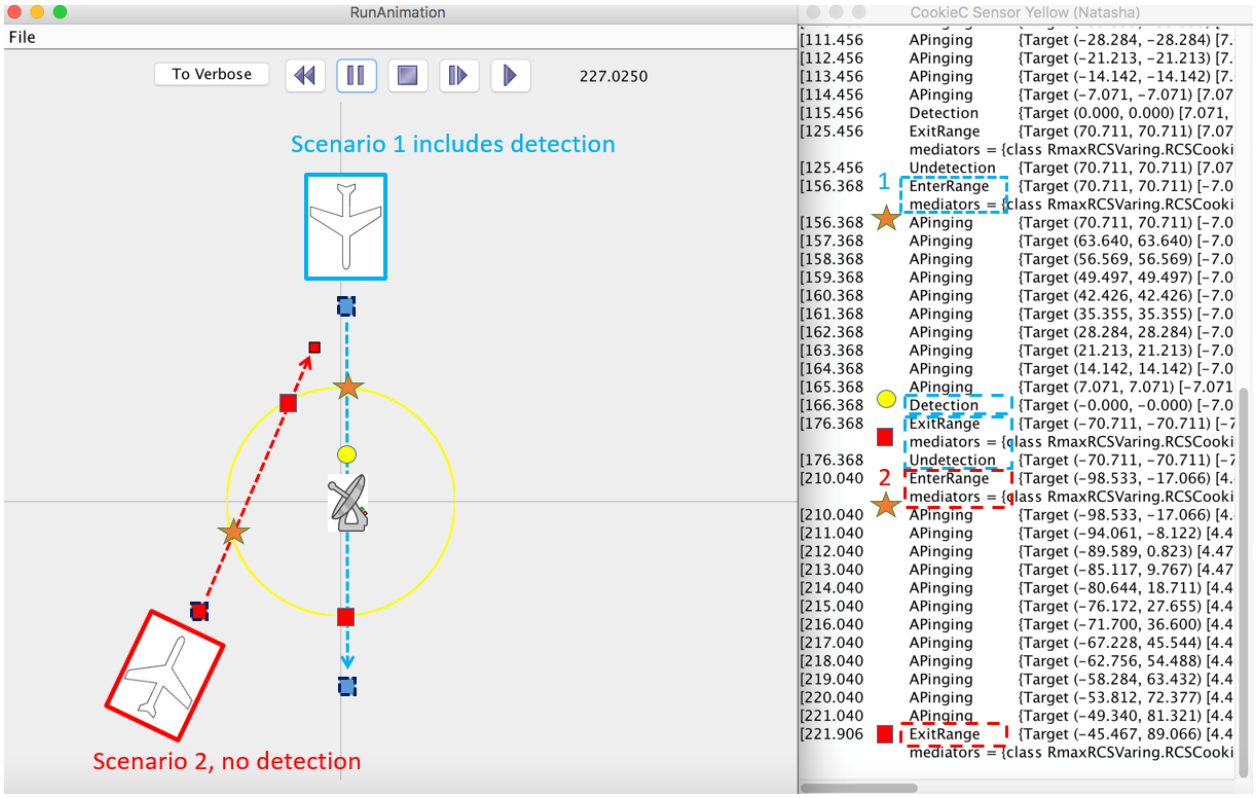


Figure 4.4. Hybrid-sensor simulation display and output produced by Java Simkit program.

4.4 Hybrid Model Simulation Results

In the portion of simulation data shown in Figure 4.5, the NOLH DPs of five parameters are shown in the blue columns, the TTD is in the yellow column, and there four additional detection information data based on the relative geometry position of target during simulation shown in the pink columns. Here the TTD results from two DoE are displayed. The top half section (red box), the target incident angle is $-0.7735165°$, while the bottom half shows the target approaching the sensor from with the incident angle = $0°$.

Waypoint_X	Incident Angle(degree)	speed(Km/hr)	Sweeping Interval(s)	prf (Hz)	thetaAz (Degree)	timeToDetect (hr)	TTD (min)	# sweep till TTD	# sweep till exit	Moving distance per ping (m)	Detection distance(km)
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.04783333	3	42	50	984.07	8.76
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.04783333	3	42	50	984.07	8.76
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.04783333	3	42	50	984.07	8.76
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.05125	3	45	50	984.07	5.86
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.05011111	3	44	50	984.07	6.82
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.05011111	3	44	50	984.07	6.82
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.05011111	3	44	50	984.07	6.82
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.04783333	3	42	50	984.07	8.76
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.04783333	3	42	50	984.07	8.76
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.04783333	3	42	50	984.07	8.76
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.05011111	3	44	50	984.07	6.82
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.04897222	3	43	50	984.07	7.78
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.04783333	3	42	50	984.07	8.76
-1.35	-0.7735165	864.06	4.1	1047.4	5.7	0.04783333	3	42	50	984.07	8.76
0	0	700	4.5	937.5	4.5	0.0475	3	38	57	875	16.75
0	0	700	4.5	937.5	4.5	0.04625	3	37	57	875	17.63
0	0	700	4.5	937.5	4.5	0.045	3	36	57	875	18.5
0	0	700	4.5	937.5	4.5	0.04375	2.4	35	57	875	19.38
0	0	700	4.5	937.5	4.5	0.04625	3	37	57	875	17.63
0	0	700	4.5	937.5	4.5	0.045	3	36	57	875	18.5
0	0	700	4.5	937.5	4.5	0.0425	2.4	34	57	875	20.25
0	0	700	4.5	937.5	4.5	0.04375	2.4	35	57	875	19.38
0	0	700	4.5	937.5	4.5	0.045	3	36	57	875	18.5
0	0	700	4.5	937.5	4.5	0.0475	3	38	57	875	16.75
0	0	700	4.5	937.5	4.5	0.0475	3	38	57	875	16.75
0	0	700	4.5	937.5	4.5	0.045	3	36	57	875	18.5
0	0	700	4.5	937.5	4.5	0.0475	3	38	57	875	16.75
0	0	700	4.5	937.5	4.5	0.04375	2.4	35	57	875	19.38
0	0	700	4.5	937.5	4.5	0.0475	3	38	57	875	16.75

Figure 4.5. Hybrid-sensor model simulation outputs: results, emphasizing time to detect (TTD), and related parameters.

The TTDs in either DoE section do not have deterministic results due to introduction of the Swerling factor, which is considered in the RRE by a Chi-Squared RV with four degrees of freedom. The TTD represents the time delay for the sensor to detect the target once the target enters the maximum detection range using units in hours.

Other detection data shown in the pink columns are the simulation information based on the relative geometric position between the sensor and targets. The definitions are as follows:

- Number of sweeps till TTD. Once the target enters the maximum detection range of the sensor, the sensor starts scanning on the target in the period of "Sweeping Interval(s)," the input parameters in cyan column. This parameter shows the number of sweeps the sensor applies on the target from when the target enters the range until the detection happens.
- Number of sweeps till exit. This parameter shows the number of sweeps the sensor applies on the target from when the target enters the range until the target leaves the

maximum detection range of the sensor. The distance that the target travels through the sensor maximum detection range by the waypoint is shown in Figure 4.6a and is calculated as follows:

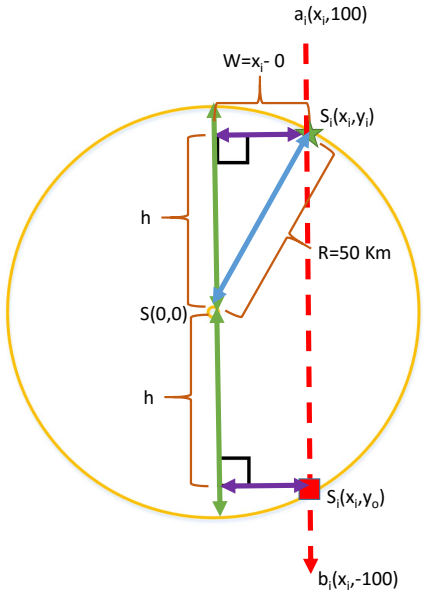
$$h = \sqrt{R^2 - W^2} = \sqrt{R^2 - x_i^2} \tag{4.3}$$

where the total distance that the target travels through the sensor is $2 \times h$.

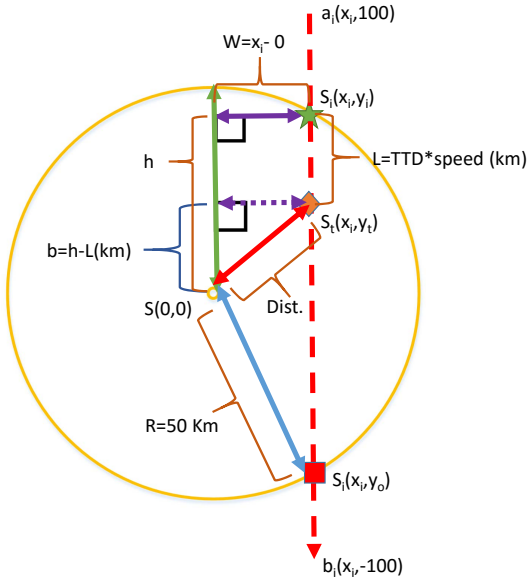
$$\text{Number of sweeps till target exits} = \frac{2h}{\text{Sweeping inteval(s)}} \tag{4.4}$$

- Moving distance per ping(m). This parameter calculates the moving distance of the target during each sweeping period of the radar in the speed designated by DoE. It is computed as follows:

$$\text{Moving distance per ping (s)} = \frac{\text{speed(km/hr)}}{60^2} \times \text{Sweeping interval (s)} \tag{4.5}$$



(a) Target's traveling distance within the sensor.



(b) The distance when the target is detected.

Figure 4.6. Target's relative position geometry to the sensor within the maximum detection range.

- The quantitative summary of the TTD of the sensor to detect targets after the targets enter the maximum detection range is shown in Figure 4.7c. The maximum TTD is 13.8 min while the minimum TTD is 2.4 min by the sensor with the maximum detection range = 50 km. The traveling distance that the target covers during the TTD depends not only on the length of TTD, but also on the speed of the target. The target traveling range within the maximum detection range is introduced next.
- Detection range km. This parameter displays the distance of the target when it is detected by the radar. In the first set of DoE (red box), the target incident angle = $-0.77°$, the detection distance is from 5.86 to 8.76 km, while the second DoE (black box) with the target incident angle = $0°$, shows the detection distance is from 16.75 to 20.25 km. The geometric illustration of detection distance is shown in Figure 4.6b and it is calculated as follows: Assume the target enters the range at $S_i(x_i, y_i)$ and gets detected after TTD at $S_t(x_i, y_t)$, the distance L that the target travels during TTD is

$$L = TTD \times speed \tag{4.6}$$

$$b = h - L \tag{4.7}$$

where h is calculated in Equation 4.3. The distance of the target when detection occurs is (red arrow):

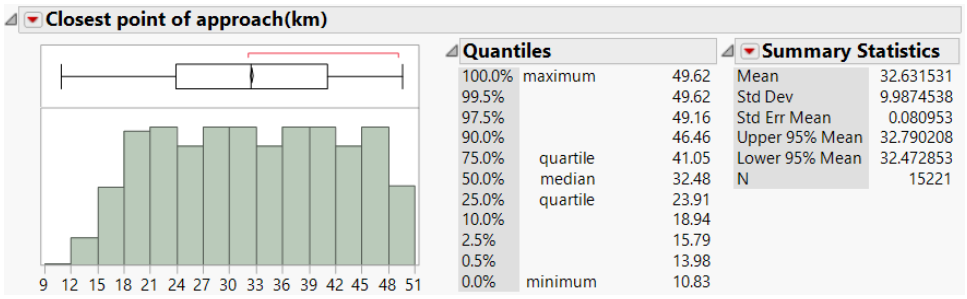
$$Dist. = \sqrt{W^2 + b^2} = \sqrt{x_i^2 + b^2} \tag{4.8}$$

Figure 4.7b summarizes the target's detected range and TTD of the target from the sensor by 6,879 out of 22,100 valid waypoints simulation, for targets that have crossed the maximum detection range of the sensor. The farthest detection distance to the target is 20.65 km, while the closest detection distance is 4.52 km. The average detection distance of this target is 10.12 km.

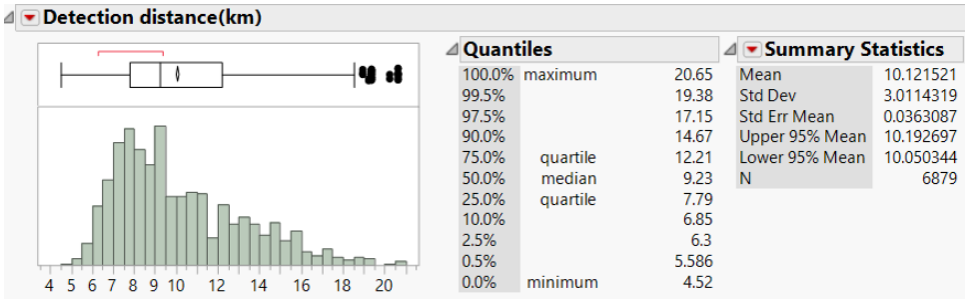
- The Closest Point of Approach (CPA) reveals the closest distance between sensor and the waypoints of the target when the sensor fails to detect the target. The CPA is shown as the length of the purple vector in Figure 4.6a and it is equivalent to the variable "waypoint_x" in Table 4.1.

  The quantitative summary of the CPA is shown in Figure 4.7a. There are 15,221 out of 22,100 valid waypoints simulation of the failure to detect the target. The maximum CPA range is 49.62 km which is about the maximum detection range of

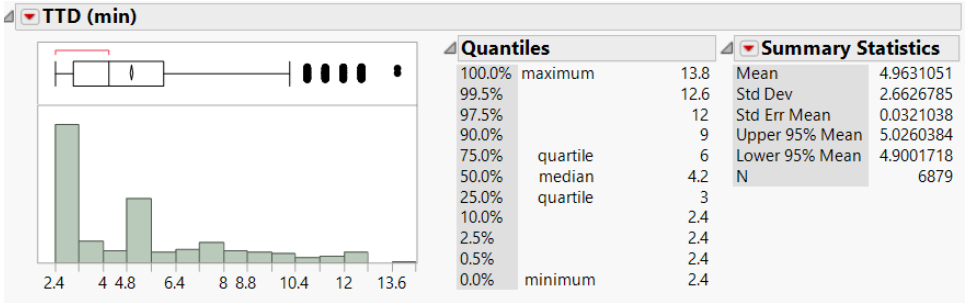
the sensor while the minimum CPA range is 10.83 km. The average CPA range for fail detection DoE is 32.63 km, which means that most detection failures happen at farther CPA range rather than the short distance. This conclusion matches the general radar detection principle that the farther the target is, the less likely the sensor detects the target.



(a) Quantitative summary of closest point of approach (km) when no detection occurs in a given replication. The maximum range of 50 km was chosen to ensure that all outlier scenarios were considered.



(b) Quantitative summary of detection range (km) when target is detected. All targets are detected within 4.52 km; no target is detected beyond 20.65 km.



(c) Quantitative summary of time to detect (min) targets. The range of TTD after targets enter the maximum detection range of the sensor is from 24 to 13.8 min.

Figure 4.7. The quantitative summary of closest point of approach (CPA) of no detection and the detection range when detection happens and the time to detect the targets after the targets enter the maximum detection range of the sensor.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Simulation Data Regression Analysis

The purpose of computing is insight, not numbers. - Richard W. Hamming [58].

5.1 Linear Regression Estimation of Time to Detect

Regression analysis is a statistical method that investigates the relationship between two or more variables related in a non-deterministic manner [59]. The relationship is expressed in the form of an equation or explanatory predictor variables. To learn the system behavior or predict the outcome associated with the input parameters of the system, the analysis process requires sufficient experiment data that cover as many detail levels in parameters as possible. Tallavajhula [60] applies data-driven methods to learn the relationship between input parameters and system state, then and constructs a high-fidelity model for a planar range sensor.

In this chapter, regression analysis is applied for fitting the meta-models that retain the equally likely behavior as the baseline hybrid sensor model, with less model structure complexity and more execution efficiency. The regression analysis is based on the observations generated by the DoE of the hybrid sensor model. Executing the DoEs through the hybrid sensor model simulation yields TTD of the target, which is associated with 255 design points with 100 replications of each DP. In this sensor, any DoE with the waypoint that falls outside of the sensor Ru is ignored. There are 221 DPs executed, yielding 22,100 simulation replications. In order to have a more concise form to represent the hybrid sensor model behavior, a meta-model is determined through linear-regression analysis for predicting the response variable "time-to-detect" the target by the sensor. The regression analysis proceeds based on the observation data generated by the DoE of the hybrid sensor model with the parameters in the range listed in Table 4.1. For a good meta-model, the goal is to minimize the difference between the predicted TTD from meta-model and the actual TTD generated from the baseline hybrid sensor model simulation. A simple linear-regression model that describes the relationship between response variable $Y$and predictor variables

$X_1, X_2, \cdots, X_p$ is defined as follows [61]:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon \tag{5.1}$$

where $\beta_0, \beta_1 \cdots, \beta_p$, called the regression coefficients, and ϵ is assumed to be a normal random error to represent the failure of the model to fit the data exactly. It is represented as follows:

$$\epsilon \sim N(0, \sigma^2) \tag{5.2}$$

These values are unknown constants to be determined from the simulation output data. Backward elimination of least-Squared error estimation is applied to form the regression meta-model. It is a commonly used technique to fit the model with minimum sum of squared errors between the meta-model and sample data. Mean Squared Error (MSE) is defined as the error between fit model and sample data. It is represented as follows:

$$\sigma^2 = \frac{SSE}{n - I} = MSE \tag{5.3}$$

where the Sum of Squared Errors (SSE) and the total number of data samples is n and I is the number of data sample groups.

5.1.1 Assumptions about the Data in Linear-Regression Model

There are three critical assumptions about the data when fitting a linear-regression model. If the data violates these assumptions, the model may give misleading or incorrect results [62].

1. **Independence:**
   The assumption may be violated when measurements are serially correlated, or when collected in batches with the instruments that have intrinsic errors. For the most common effects caused from the dependency data, the result is to underestimate true variability, which leads to a higher probability of Type I error (false rejection of the null hypothesis).
2. **Normality :**
   The data is assumed to be normal distributed. It may increase Type II errors when the data distribution has heavy tails, making it prone to producing outliers.

3. **Homoscedasticity :**
   This assumes data has equal-variance across treatments, while the unequal-variance of data is called heteroscedasticity. Heteroscedasticity often goes with non-normality. The typical effect is to make more Type I errors in the estimation.

5.1.2 Measuring Goodness of Fit

To evaluate the level of fitness between the predicted data from the meta-model to the actual data in regression analysis, two measurement units are introduced in the regression analysis to represent the goodness of fit in the meta-models:

1. **Residuals:**
   Residuals are the errors made in using the regression predictions in place of the actual y-values (the actual data):

$$\hat{\varepsilon}_i = y_i - \hat{y}_i \tag{5.4}$$

   where y_i denotes the actual data, $\hat{y}_i$ is the predicted data from meta-model and $\hat{\varepsilon}_i$ is the residual of predicted y_i.
   In matrix notation:

$$\hat{\varepsilon} = y - X\hat{\beta} \tag{5.5}$$

   where $\hat{y}$ denotes the actual data in n-vector, $X\hat{\beta}$ is the predicted $\hat{y}$ matrix, and $\hat{\varepsilon}$ is the residuals in n-vector of predicted data. Figure 5.1 illustrates the relationship of residuals and the predicted model. The blue surface is the predicted response parameter $\hat{y}$, which consists of the predictor parameters x_i and the predicted coefficient β_i. The residuals are the error between the predicted response $\hat{y}_i$ and the actual data y_i. The summation of the square of residual vector provides an idea of how well the regression prediction $\hat{y}$ can approximate the value of the response variable y [63]. The Residual Sum of Squares (RSS) is defined as:

$$RSS = \sum_{i=1}^{n} \hat{\varepsilon}_i^2 = \boldsymbol{\varepsilon}^{\boldsymbol{T}} \cdot \boldsymbol{\varepsilon} = ||\hat{\varepsilon}_i^2|| = ||y - \hat{y}||^2 \tag{5.6}$$

   and the estimated variance $\hat{\sigma}^2$ of the residual ε_i is defined as:

$$\hat{\sigma}^2 = \frac{RSS}{n - I} \tag{5.7}$$

2. **RSquared value :**

   However, the RSS is hard to interpret by itself since it has its own measurement unit as y. Thus a unitless quantity, "coefficient of determination" or "proportion of variance explained," is converted from RSS divided by Total Sum of Squares (TSS). It is denoted as R^2 and read as "RSquared Value". R^2 value is a useful measurement to indicate how well the model fits the data, and it is in the range between zero and one. It is defined as:

   $$R^2 = 1 - \frac{RSS}{TSS} \tag{5.8}$$

   while

   $$TSS = \sum_{i=1}^{n} (y_i - \overline{y}) \tag{5.9}$$

   $R^2 = 0$ implies that the regression was completely ineffective in improving prediction of y, while $R^2 = 1$ indicates the regression model perfectly predicts the data y. A fairly high R^2 regression model seems preferable to be able to successfully represent the sample data behavior. However, there is no fixed standard value to indicate the acceptable R^2 value, and it does not guarantee the high R^2 value model behaves more accurately than the lower value model as expected. Unless the verification methodologies have been applied and proven the meta-model has the consistent ability to predict the results. Otherwise, the model may have just fit on insignificant factors from the data without presenting the truly significant factors properly.

5.1.3 Multi-Degree Interaction Predictors

A regression model is classified as linear if $E[Y|X = x]$ is a linear function of the predictor parameter coefficients β_i, not of the predictor parameters x_i. Hence, the predictor parameters can have multiple degrees of interaction terms, and convert computation terms fit in the linear meta-model for possible better fitting behavior of the data. There are three methodologies that may be applied in this study:

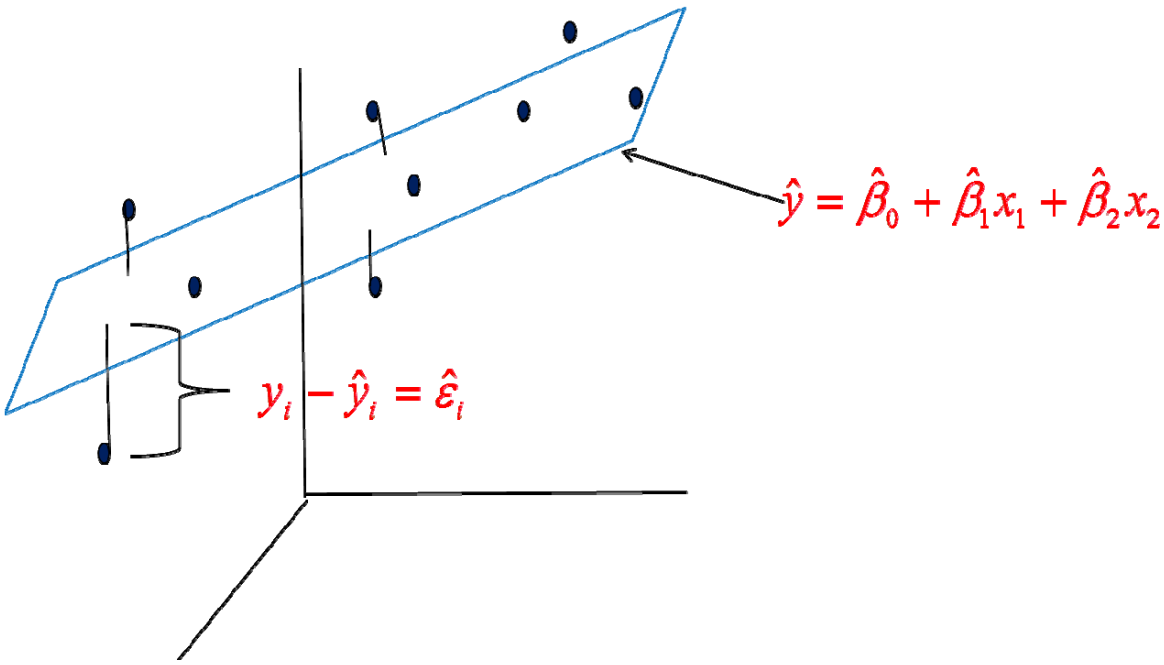


Figure 5.1. Residual differences between fitting model and actual data.

Multi-Order Polynomial Parameters

The different order of polynomial predictor parameters can be applied in the fitting model. For an up to three degree of polynomial predictor parameter x, the model is represented as:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \tag{5.10}$$

Multi-Order Factorial Parameters

The different order of interaction terms between predictor parameters can be applied. For a model with up to three degree of factorial predictor parameters, the model can be represented as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1 x_2 + \beta_3 x_2 x_3 + \beta_4 x_1 x_3 + \beta_5 x_1 x_2 x_3 \tag{5.11}$$

Predictor Parameters Conversion

Mathematical converting operations can be applied on the predictor parameters and fit in the model such as logarithmic conversion and trigonometric functions. The fitting model can be represented as:

$$y = \beta_0 + \beta_1 log(x_1) + \beta_2 sin(x_2) + \beta_3 cos(x_2) sin(x_3) ... \tag{5.12}$$

5.1.4 Model Selection Methodology

Often the correct model is unknown when fitting a regression model. Therefore the purpose of the regression analysis includes figuring out what are the values of the regression coefficients and which predictor variables belong in the regression model. The RSquared value always has monotonic improvement as a variable and is included in the model. However, the improvement does not necessarily mean the fit model is closer to the real model. Introduced here are several model selection methodologies and criteria.

Least-Squared Error

A mathematical procedure for finding the best-fitting model to the sample data minimizes the SSE of the offsets ("the residuals") of the sample data from the fitting curve. The linear least-squares fitting technique is the simplest and most commonly applied form of linear regression, and often provides a solution to the problem of finding the best fitting model through a set of data [64].

Forward Selection

The model fitting procedure is:

- Start with no predictors in the model.
- Add the predictor that would have the smallest p-value and less than the threshold $\alpha = 0.05$.
- Keep the selection operation one predictor at a time until no more predictors meet the selection threshold [63].

Backward Elimination

The selection procedure is opposite to the "Forward Selection":

- Start with all predictors of interest in the model. The model can start from "Least-Squared Error" models or models which have been fit by other selection methods that may still have been overfitting.
- Remove the predictor that would have the largest p-value and larger than the threshold $\alpha = 0.05$.
- Keep the removal operation one predictor at a time until no more predictors meet the elimination threshold.

Step-wise Selection

The model is built by the hybrid of forward selection and backward elimination selection principle with the designated parameter selection rules. The models are fit by alternating the selection and elimination steps. The typical predictor selection rule is by p-value threshold. The alternating model selecting steps keeps going until no new predictor can meet the p-value threshold for adding and no existing predictor can meet the p-value threshold for elimination.

In the adding or subtracting process of predictor selection, the higher the p-value of the predictors, the less likely to reject the null hypothesis. In other words, the high p-value predictors represent the insignificant predictors in the model since these terms statistically make no difference in the meta-model.

On the contrary, low p-value predictors represent the significant predictors in the meta-model since the model containing the new predictor statistically performs differently than the model without the new term.

Akaike Information Criterion

When the model is fit with more predictors, it brings down RSS monotonically. This criterion introduces penalty terms for the evaluation value as the number of predictors included. The Akaike Information Criterion (AIC) is computed as follows [65]:

$$AIC = n \log\left(\frac{RSS}{n}\right) + 2p \tag{5.13}$$

where p is the number of predictor parameters that have been included in the model and n is the total number of observation data. This criterion can be applied to the stepwise selection method for the model selecting stop point criteria. The preferred model is chosen with the smallest value of AIC among those under consideration. For large values of q, there may be a large number of predictors included, which can lead to overfitting models. The p factor raises AIC for the model such that the complexity becomes too great.

Bayes Information Criterion

The Bayes Information Criterion (BIC) is defined as follows:

$$BIC = n \log\left(\frac{RSS}{n}\right) + p \log(n) \tag{5.14}$$

This criterion replaces the $2p$ in AIC with the $p \log(n)$ and tends to prefer smaller models to the AIC. The BIC penalizes adding another predictor more than AIC does. This results in smaller fit model. [66]

5.1.5 Fitting a Linear-Regression Model

The regression analysis is based on the DoE simulation scenario data in section 4.2. Executing the simulations on the hybrid sensor model with five factors DoE, 256 NOLH DPs, and each DP runs 100 replications respectively, a linear-regression analysis is applied on the hybrid sensor model simulation results with the response parameter "TTD." Because fitting a regression model with predictors terms of different complexity produces different predicted TTD response models, there is no simple rule for model selection and no quick answer for judging the fidelity of the fitting meta-model. There are two meta-models are fitted for predicting TTD with different complexity of predictors parameter interaction

terms in this section, including analysis of the validity of the predicted data from these two meta-models.

Fitting Model with Interaction Terms up to Degree of Two

Applying the predictor parameters listed in Table 4.1, up to two degrees of factorial and polynomial terms in a fit model allow for capture of curvatures and distribution behavior in the sample data. In order to fit a meta-model with proper terms, and avoid underfitting or overfitting the model, there is a two-stage process for choosing the predictor terms for the linear-regression model:

1. Forming a linear model by the method of the least-squared error approach. This method generates the meta-model with the predictors that have the least-squared error with the analysis data. This method should prevent the underfitting model based on the all available predictors since the least error model has been found in the process.
2. Applying "backward elimination" to the meta-model. The underfitting model is not a preferable result. In contrast, the overfitting model also needs to be avoided since the more insignificant predictors are kept, the more likely the model will present insignificant and irrelevant results. These results are the misleading information from the overfitting meta-model. Thus, after forming the meta-model by the least-squared error approach, each predictor parameter has a p-value to indicate how significantly the factor contributes to the model. The backward elimination approach is applied to the least-squared error model by setting the significant p- value = 0.05 as the threshold for selecting the predictor terms. The elimination starts from the term with the highest p-value and then rebuilds the model after the elimination. Repeat eliminating the highest p-value term that is over 0.05 one by one, until there is no predictor term p-value larger than the significant value 0.05.

This meta-model has RSquared = 0.966. A higher RSquared value indicates more response data around the baseline model mean.

$$\sigma^2 = \frac{SSE}{n - I} = MSE = 0.008167 \tag{5.15}$$

where the sum of squared errors is SSE, the total number of data samples is n=6878, and

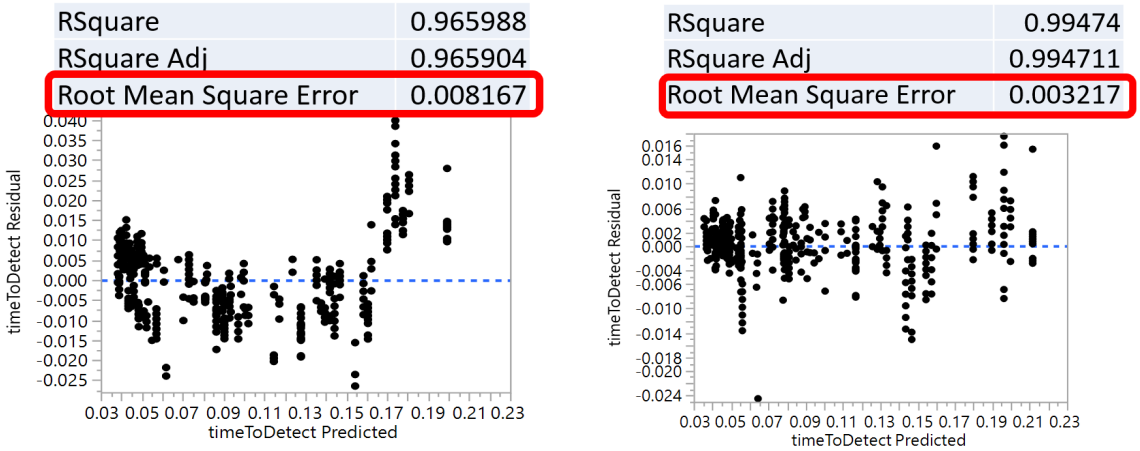
the number of data sample groups is I = 39 in this meta-model.
Figure 5.2a shows the residual of fit model by predicted "time-to-detect (TTD)." The distribution of the residual of the meta-model shows in a concave-up trend which makes the estimation of prediction error term misrepresented by the normal RV. The distribution of the prediction data residual tends to be normally distributed. In addition, certain heteroscedasticity can be observed in residual distribution. The variance of residuals increases gradually along with the increment of the predicted TTD. This means unequal-variance across treatments that is not the ideal assumption for forming a meta-model by linear regression by degree of two interaction terms of predictor variables.

Fitting Model with Interaction Terms up to Degree of Four

The two-degree interaction terms meta-model does not seem to have enough expressive power to represent the behavior of the hybrid model well. Increasing the complexity of fitting the model with a higher order of interaction terms of predictor parameters allows the meta-model to capture more complex curves in the simulation replication data distribution. There exists another regression model with higher complexity of interaction terms of predictor parameters up to four degrees of factorial and polynomial terms in a fit model. Following a similar model selection process as the previous two-degree interaction terms model, the meta-model has RSquared = 0.9947 with the estimated MSE = σ^2 = 0.003217. Figure 5.2b shows the residual of fit model by predicted "time-to-detect (TTD)" with a much better information pattern shown in the predicted TTD residuals distribution than the two-degree interaction terms model. Certain heteroscedasticity can be observed in the residual distribution, but is much less obvious than in the previous low complexity meta-model. This may be a more preferable meta-model to represent the hybrid sensor model.

The overall meta-model expression is described in Appendix C.1. However, "all models are wrong, but some are useful," as observed by George Box. There will not be a perfect model to represent the original model completely, thus the regression models with different complexity of predictors can be created, and more evaluation methods for the fit models are applied in further checking for fidelity and model selection. Furthermore, since the "data" being fit is actually the metrics corresponding to individual simulation and replications, interesting questions arise regarding inputs, outputs, and fundamental usefulness of the

curve-fit characteristics. Further future investigations may prove worthwhile.



(a) Fit model residual with predictors to degree of 2.(b) Fit model residual with predictors to degree of 4.

Figure 5.2. Residual plot by predicted time-to-detect (TTD) of fit model with the predictor parameters in factorial and polynomial interaction terms in degree of 2 and degree of 4.

5.2 Logistic-Regression of P_d

For a dichotomous dependent variable, such as if the target is detected, the numerical value of the variable is not intrinsically interesting. The key point to focus on is whether the classification of cases into one or another categories of the dependent variable can be predicted by the independent predictor variables [67]. The probability of an event happening is denoted as $P(Y = 1)$. If the probability of an event happening is known, then the probability of an event not happening is also known: $P(Y = 0) = 1 - P(Y = 1)$. Logistic-regression analysis models the ratio between $P(Y = 1)$ and $P(Y = 0)$ with a natural logarithm of the ratio. This expression is called logit of Y:

$$ln\left\{\frac{P(Y=1)}{[1-P(Y=1)]}\right\} = logit(Y) = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \tag{5.16}$$

then the $logit(Y)$ in Equation 5.16 can be converted back to the probability that (Y=1) with predictor variables in such equation:

$$\text{Probability}(Y=1) = P(Y=1) = \frac{1}{1+e^{-(\alpha+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_p X_p)}} \tag{5.17}$$

Y represents "no detect" event is this study; therefore, the probability of no detection of the sensor to the target is calculated by:

$$\text{Probability(NoDetect=1)} = \text{Probability}(Y = 1) = P(Y = 1) = \frac{1}{1 + e^{-logit(Y)}} \tag{5.18}$$

The probability of detection of the target is represented by the "no detect" event is false, and it is calculated by:

$$\begin{aligned}
&\text{Probability(Detection)} \\
&= \text{Probability(NoDetect=0)} \\
&= P(Y = 0) = 1 - P(Y = 1) \\
&= 1 - \frac{1}{1 + e^{-logit(Y)}} \\
&= \frac{\left[1 + e^{-logit(Y)}\right] - 1}{1 + e^{-logit(Y)}} \\
&= \frac{e^{-logit(Y)}}{1 + e^{-logit(Y)}} \\
&= \frac{\frac{e^{-logit(Y)}}{e^{-logit(Y)}}}{\frac{1}{e^{-logit(Y)}} + \frac{e^{-logit(Y)}}{e^{-logit(Y)}}} \\
&= \frac{1}{\frac{1}{e^{-logit(Y)}} + 1} \\
&= \frac{1}{1 + e^{logit(Y)}}
\end{aligned} \tag{5.19}$$

5.2.1 Procedure for Fitting Logistic-Regression Model

A logistic-regression analysis is applied on the hybrid sensor model simulation described in section 3.3. The DoE is based on a five factors, 256 NOLH DPs, and each DoE runs 100 replications respectively. In the scenario, the sensor is set to the maximum detectable range Ru = 50 km. When targets are outside this maximum range, the sensor does not process the detection algorithm. This DES structure isolates the irrelevant events outside of the significant information region. The logistic-regression analysis predicts the response variable P_d of the sensor detects the target, using predictor parameters of up to a

degree of two polynomial and factorial terms. Since not all the predictor parameters and interactive predictors contribute a significant effect on the predicted meta-model during the regression process, the meta-model should only contain significant predictors in the meta-model. There is a similar two-stage process for choosing the proper predictor terms for the logistic-regression model as shown earlier for the linear-regression model selection process.

1. Forming a logistic model by the method of least-squared error approach. This method generates the meta-model with the predictors that have the least-squared error with the analysis data for preventing the underfitting model.
2. Applying "backward elimination" to the logistic meta-model. This process eliminates the insignificant predictors by the threshold p-value = 0.05. The predictors, which have the largest p-value and greater than the threshold value get removed one at a time until there are no insignificant predictors remaining.

5.2.2 Logistic-Regression Model Representation

This logistic-regression analysis has an RSquared = 0.9376. The logistic meta-model determines if the target is detected by the sensor in this DoE with a predicted probability. Figure 5.3 shows part of the DoE that is applied on the hybrid model. The results from the model indicate whether a detection happens in this DoE in this replication, and if the detection happens what the TTD of the sensor to the target in this simulation. The red box data in Figure 5.3 show that no detection occurs in these DoE replications. The "NoDetection" column presents if the detection occurs from the hybrid model simulations. When "NoDetection"=1, it indicates that no detection occurred. When "NoDetection"=0, it indicates that a detection occurred (the blue box data). The predicted P_d is shown in the "ProbOfDetection" column and is represented and calculated in the form of Equation 5.19 and the logit function in Equation 5.16 is described with all detailed predictor terms in Appendix C.5. The "Most Likely NoDetection" prediction is determined by the Probability$(Y = 0) = P(Y = 0)$ in Equation 5.20 to predict the P_d of the sensor to the target, and it is represented as follows:

Incident Angle(degree)	speed(Km/hr)	Sweeping ltg(s)	prf	thetaAz	NoDetection	ProbOfDetection	Most Likely NoDetection	timeToDetect (hr)	Pred Formula timeToDetect
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-23.103917	793.75	1.8	1003.4	5.9	1	5.09491e-35	1	•	-0.763004692
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.13088889	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.13088889	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.13088889	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274
-6.2173273	360.16	3.8	1086.9	5.2	0	0.9998310966	0	0.12983333	0.1294221274

Figure 5.3. Partial section of DPs of hybrid sensor model simulation TTD results with regression analysis predictor and response variables.

$$\text{Most likely no detection} = \begin{cases} 0, & \text{Probability(Detection)} < 0.5 \\ 1, & \text{Probability(Detection)} > 0.5 \end{cases} \tag{5.20}$$

In the simulation scenario based on the regression meta-model, the target detection can be determined by comparing predicted P_d with a uniform random variable [0,1] in a DoE scenario.

$$\text{If detection} = \begin{cases} 0, & \text{Probability(Detection)} < \text{RV} \sim \text{Uniform } [0,1] \\ 1, & \text{Probability(Detection)} > \text{RV} \sim \text{Uniform } [0,1] \end{cases} \tag{5.21}$$

If the target detection is determined to be true by comparing the P_d with a uniform random variable [0,1], then time-to-detect the target can be estimated by the linear-regression model described in section 5.1.

5.3 Shaping the Regression Model

Recall from the definition of the regression analysis in section 5.1.1, the linear-regression analysis is performed under two assumptions regarding variance of the observation data:

- Normality
- Homoscedasticity

In many cases, the observation data do not perfectly conform to these assumptions for regression analysis. Osborne's study [68] shows the significant violation of either assumption can increase the chances of committing either a Type I or II error depending on the nature of the analysis and violation of the assumption. The research tests these assumptions and reports how to correct for violation of these assumptions. For improving the fitness of data and correction of violations of linear-model assumptions, the operation that shapes the regression-analysis parameters can be applied. There are two kinds of parameters shaping operation introduced in this study.

5.3.1 Transformation on the Predictor Variables: Incident Angle of the Target

The meta-model is fit with the interaction terms of predictor parameters up to four degrees of factorial and polynomial terms. In the waypoint predictor parameter expression, instead of applying "Incident Angle (degree)" as described in section 5.1.5, this model converts the predictor variable "Incident Angle" into "Waypoint_x." The "Waypoint_x" represents the range of target waypoints in the x-axis component that is illustrated in section 4.2. Figure 5.5a shows the meta-model MSE=0.003228, which is similar to the model without target incident angle transformation. The model is fit with the predictors in 32 degrees of freedom and the full model prediction expression is provided in Appendix C.2.

5.3.2 Transformation of the Prediction Response Variable: Box-Cox Transformation

For improving normality, the Box-Cox transformation represents a family of powerful transformations that help researchers find the optimal normalization transformation for the variable. The transformation represents a potential best practice where normalizing data or equalizing variance is desired [69].

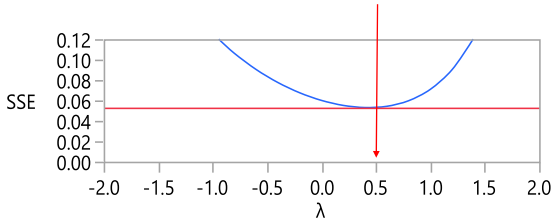


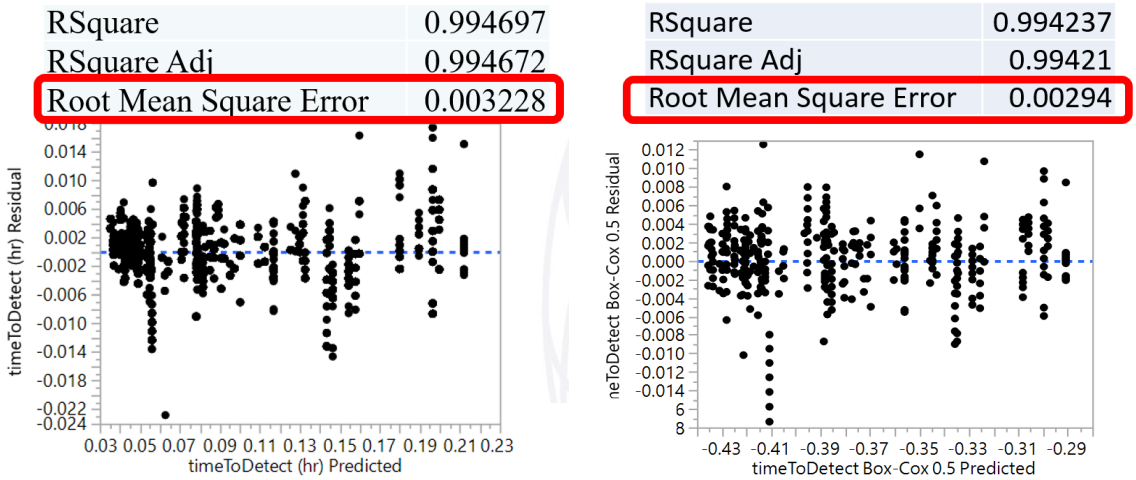
Figure 5.4. Box-Cox transformation of TTD prediction response variable with lowest SSE $\lambda = 0.4$.

The Box-Cox transformation is usually applied when the response variable Y is non-negative and takes a parameter λ. The definition is as follows [70]:

$$g_\lambda(y) = \begin{cases} \frac{y_\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(y), & \lambda = 0 \end{cases} \tag{5.22}$$

To choose the best λ for the transformation of the response variable Y, the transformation computes the SSE_λ with $\lambda[-5, 5]$ of all the response variables $g_\lambda(y)$ that have been transformed by all different λ values by Equation 5.22. In Figure 5.4 shows the Box-Cox transformation applied to the predicted response variable TTD. The good transformation results can be visually observed about $\lambda = 0.5$ event if $\lambda = 0.4$ is the lowest point by the computational report.

A comparison of the model before the transformation in Figure 5.2b with the model after Box-Cox transformation with $\lambda = 0.5$ is shown in Figure 5.5b; the heteroscedasticity of variance in the residual distribution has certain improvement after transformation. The model applied to the transformation has MSE=0.00294, which is less than the one before the transformation with the MSE = 0.003217. The results show that the model predicting response has been positively modified in a certain sense after applying the Box-Cox transformation with less MSE.



(a) Fit Model with predictors "Waypoint_X." (b) Fit Model with predicted TTD applied to Box-Cox.

Figure 5.5. Residual Plot by Predicted Time to Detect (TTD) of Fit Models with the predictor parameters in factorial and polynomial interaction terms in degree of 4. (a) applies the predictor conversion of waypoint from "Incident Angle(degree)" converted into "Waypoint_X." (b) applies the response variable TTD with Box-Cox transformation by $\lambda = 0.5$.

Box-Cox Transform Limitation

1. The transformation is not a guarantee for normality. It is because the transformation operation does not really check the normality of the response parameters. This method checks for MSEs and assumes the λ that creates the smallest MSE of transformed $g_\lambda(y)$ has the best result [71].
2. The Box-Cox transform only works if all the data are positive. Non-positive data can be offset by adding a constant that brings all data to be positive values before the transformation process.
3. The transformation creates the biased estimator, and it is possibly difficult to convert the transferred estimator back to the original response estimator. For a simpler converting operation, choose the most conventional-looking λ in the CI. In this study, $\lambda = 0.5$ Box-Cox transformation for TTD estimator y by Equation 5.22 is chosen to avoid the complex conversion process, rather than choosing the lowest $\lambda = 0.4$. The transformation is computed as follows:

$$g_\lambda(y) = \frac{y^{0.5} - 1}{0.5} = 2y^{\frac{1}{2}} - 2 \tag{5.23}$$

The expected value of the transferred TTD estimator $g_\lambda(y)$ is computed as follows:

$$\begin{aligned} E\left[g_\lambda(y)\right] &= E\left[2y^{\frac{1}{2}} - 2\right] \\ &= 2E\left[2y^{\frac{1}{2}}\right] + E[-2] \\ &= 2E\left[2y^{\frac{1}{2}}\right] - 2 \end{aligned} \tag{5.24}$$

By definition:

$$\begin{aligned} & E\left[X^2\right] = E^2\left[X\right] + \sigma_X^2 \\ & \Rightarrow E^2\left[X\right] = E\left[X^2\right] - \sigma_X^2 \\ & \Rightarrow E\left[X\right] = \pm\sqrt{E\left[X^2\right] - \sigma_X^2}, \text{ assume } X = y^{\frac{1}{2}} \\ & \Rightarrow E\left[y^{\frac{1}{2}}\right] = \pm\sqrt{E\left[\left(y^{\frac{1}{2}}\right)^2\right] - \sigma^2_{y^{\frac{1}{2}}}} \\ & \Rightarrow E\left[y^{\frac{1}{2}}\right] = \pm\sqrt{E\left[y\right] - \sigma^2_{y^{\frac{1}{2}}}} \end{aligned} \tag{5.25}$$

Substitute $E\left[y^{\frac{1}{2}}\right]$ from Equation 5.25 into Equation 5.24, then the expected value of transferred TTD estimator is:

$$E\left[g_\lambda(y)\right] = 2 \pm \sqrt{E\left[y\right] - \sigma^2_{y^{\frac{1}{2}}}} - 2 \tag{5.26}$$

Equation 5.26 has shown that the Box-Cox transformation creates a biased estimator that cannot be directly applied for the estimation.
The transferred estimator has to be inversely transferred back to the original estimator. The inverse transformation of TTD estimator is computed as follows:

$$\begin{aligned} & 2y^{\frac{1}{2}} = g_\lambda(y) + 2 \\ & \Rightarrow y^{\frac{1}{2}} = \frac{1}{2}g_\lambda(y) + 1 \\ & \Rightarrow y = \left(\frac{1}{2}g_\lambda(y) + 1\right)^2 \\ & \Rightarrow y = \frac{1}{4}g_\lambda(y)^2 + g_\lambda(y) + 1 \end{aligned} \tag{5.27}$$

5.4 Meta-Model Verification

The RSquared value is one of the factors that indicates how well fit the meta-model to the hybrid sensor model. The four-degree polynomial and factorial interaction predictors are given in section 5.1.5; the RSquared of this meta-model is 0.9947, which means the meta-model is highly correlated to the sample data that is generated from the DoE.

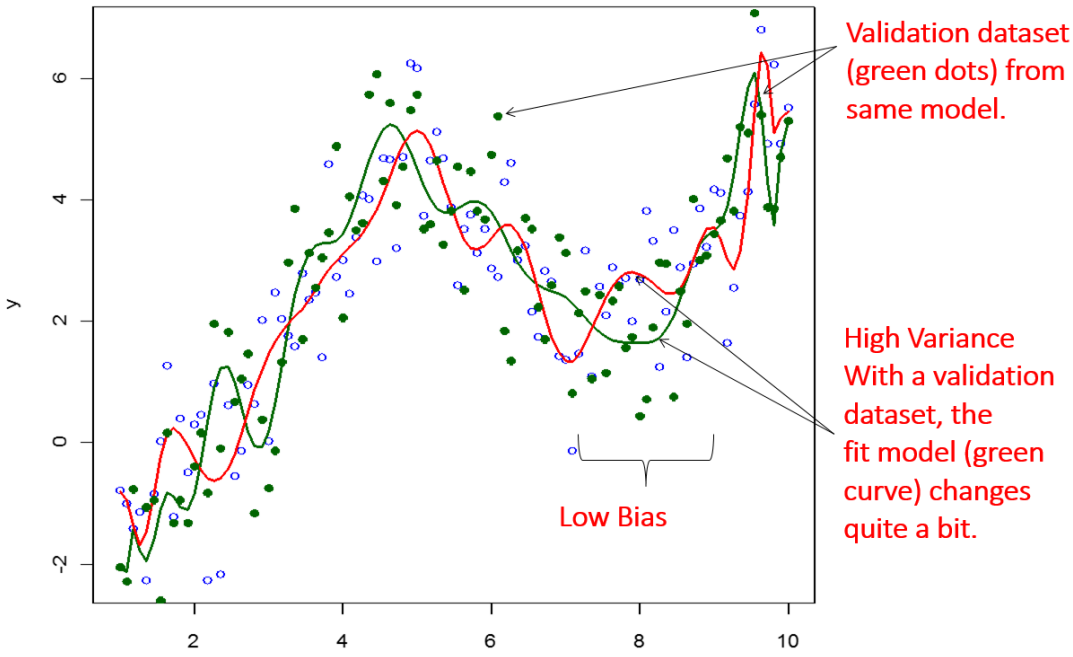
The RSquared value indicates the goodness of fit of the meta-model to the sample data. One factor about the relationship between the complexities of the fitting model to the RSquared value of the model is that the RSquared value initially increases with the increment of the complexity of the fitting model. The model might eventually reaches an RSquared value almost equal to one, which means the fitting model perfectly match the sample data by adding more predictor parameters to the meta-model. However, does the meta-model represent the right data trend or the model actually captures the insignificant information from the observation data? Therefore, the further model verification methodologies are applied for examining if the meta-model performs correctly.

5.4.1 Cross-Validation

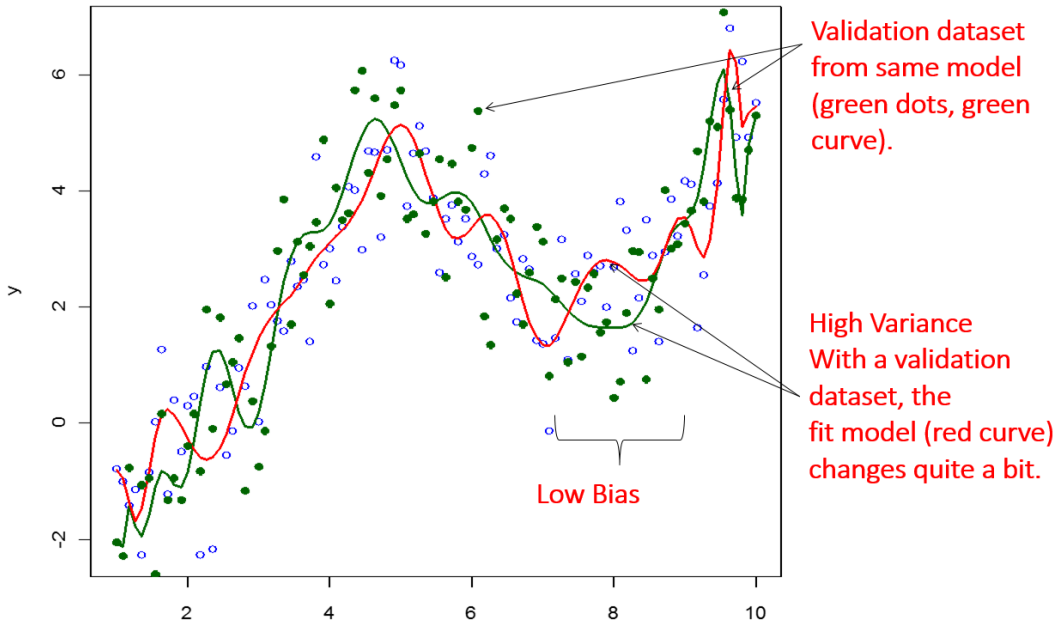
If there are unlimited number of examples for the analysis of the model available, the answer to selecting a right model to fit the data becomes straightforward. Choosing the model that provides the least error rate, i.e., the highest RSquared value, is the best model to represent the entire population data. However, in real-world applications, it is always a finite set of examples available for the analysis. In most situations, the amount of accessible data can be small due to the difficulty or expense of conducting more experiments [72]. Curiously for this work, since the replication data can be increased by adding more simulations with greater diversity of input parameters, an even-broader relationship exists. Nevertheless, the computer-based simulation in this study, which makes executing the simulation cheap, still has a finite number of DoE available. Thus the highest RSquared Value is no longer the only threshold for selecting the proper fitting model; since there a question remains whether the meta-model fits the most significant factors from the data, or the model just fits by the outliers that are insignificant behaviors of the data i.e., an overfitting model. Careful analysis is necessary.

There are two kinds of undesirable complexity of fitting model that need to be avoided:

- **Underfitting model:**
  The model formed with too few predictor factors has insufficient ability to predict the sample data behavior. Figure 5.6a shows an underfitting model that is fit by the training dataset (the blue circles). Due to the lack of complexity of the model fit line (red line), the model does not quite capture the trend of the sample data (blue circles). Therefore, the model has high bias from the predicted results to the sample data. When introducing another independent validation dataset (green dots) generated by the same sample model, the low complexity model fit line that is generated from the validation dataset almost overlaps with the fit line from the training dataset. This is because the training dataset and the validation dataset have the same distribution trend. Thus the underfitting model has low variance for predicted results.
- **Overfitting model:**
  The model contains too-many complex interaction predictor parameters. The model is sensitive to capture every detailed variation found in the sample data; therefore, the results show low bias between predicted results and sample data in Figure 5.6b. However, the overfitting model may just capture much insignificant information out of the sample data which are not meaningful for the sample model. In comparison to the fit models (red line, blue circles) from training data and independent validation data (green line, green dots) generated from the same sample model, there is high variance between the estimation of two sample data. Thus the estimating result from a overfitting model has high variance.



(a) Underfitting model with high bias and low variance.



(b) Overfitting model with low bias and high variance.

Figure 5.6. The graphical illustration: (a) Underfitting model, (b) Overfitting models with the analysis of bias of mean and variable. The comparison is shown by the fitting line of the training dataset (blue circles and red curve) and the validation dataset (green dots and green curve). Source: [73].

Two-way Data Splits

In order to verify whether the models are fit with the proper number of predictor parameters, the CV needs to be applied. CV is the methodology to verify whether the meta-model fits with the proper complexity of predictor data by dividing data into different portions for training the meta-model and testing the error rate of the model. In the two-group method, all available data are split into two datasets:

- **Training Set:**
  A training set takes the majority of the data. Usually it contains around 70 percent of available data and it is used to train the regression model for predicting the sample data. The ratio of retaining data is dependent on the amount of available data. In the sparse sample data case, the training dataset may need to retain a larger portion of all the data in order to have sufficient data to train the regression model properly. Whereas, if the dataset is abundant, the smaller portion of training set data is sufficient to train the model properly. It leaves out more available data for verification analysis.
- **Validation Set:**
  The validation set is used to estimate the error rate of the trained regression meta-model that is produced by fitting the independent training set data, which are generated from the same sample model.

There are two goodness-of-fit measurements applied for evaluating the error rate of the training meta-model by the training dataset and the validation dataset [74]:

- **RSquared Value Validation :**
  This is similar to the RSquared value measurement introduced in section 5.1.2 for representing the error between predicting data and observation data. The difference of RSquared Value Validation, which is denoted as $RSquared\ Validation$, is that the prediction error is computed from each observation in the validation dataset instead of from the training dataset. This is the difference between the actual response and the response predicted by the training dataset model. Thus the properly fit meta-model that is trained by the training dataset is expected to have a closer error rate as the model is verified with the independent validation dataset. In contrast, a wrongly fit model is expected to have a larger error rate in the validation process. It is represented

as follows:

$$RSquared\ \ Validation = 1 - \frac{SSE_{Validation}}{SST_{Validation}} \tag{5.28}$$

  - Square and sum the prediction error to obtain $SSE_{Validation}$.
  - Square and sum the differences between the actual responses in the validation set and their mean. This is the $SST_{Validation}$.

  The higher the RSquared Validation Value, the lower the error rate to the predicted data to the sample data.

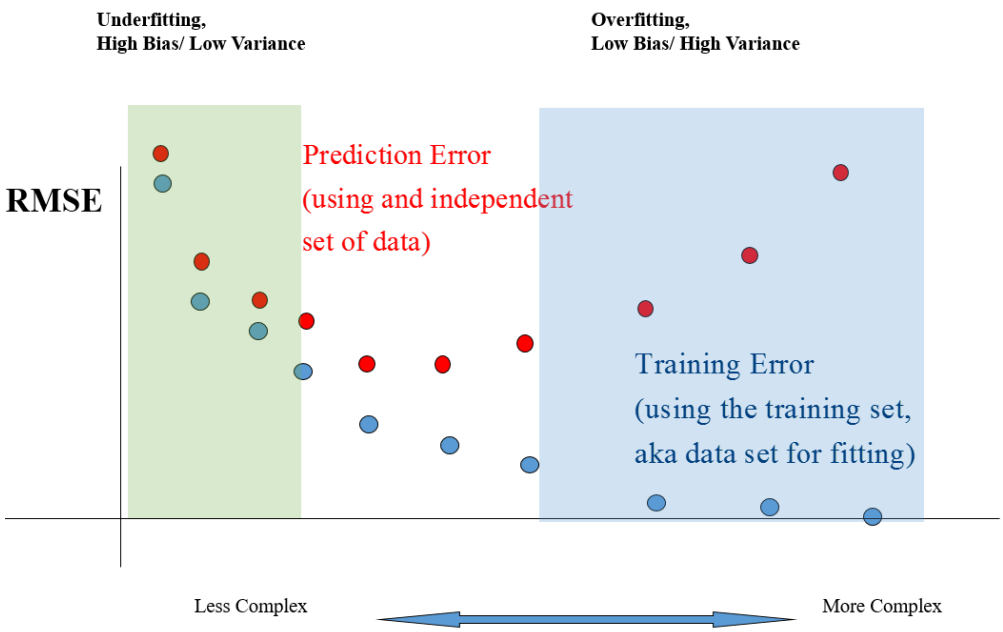
- **Square Root of the Mean Squared Prediction Error (RMSE) Validation:**
  The square Root of the Mean Squared prediction Error (RMSE) for the validation dataset is computed as follows:

$$RMSE\ \ Validation = \sqrt{\frac{SSE_{Validation}}{n_{Validation}}} \tag{5.29}$$

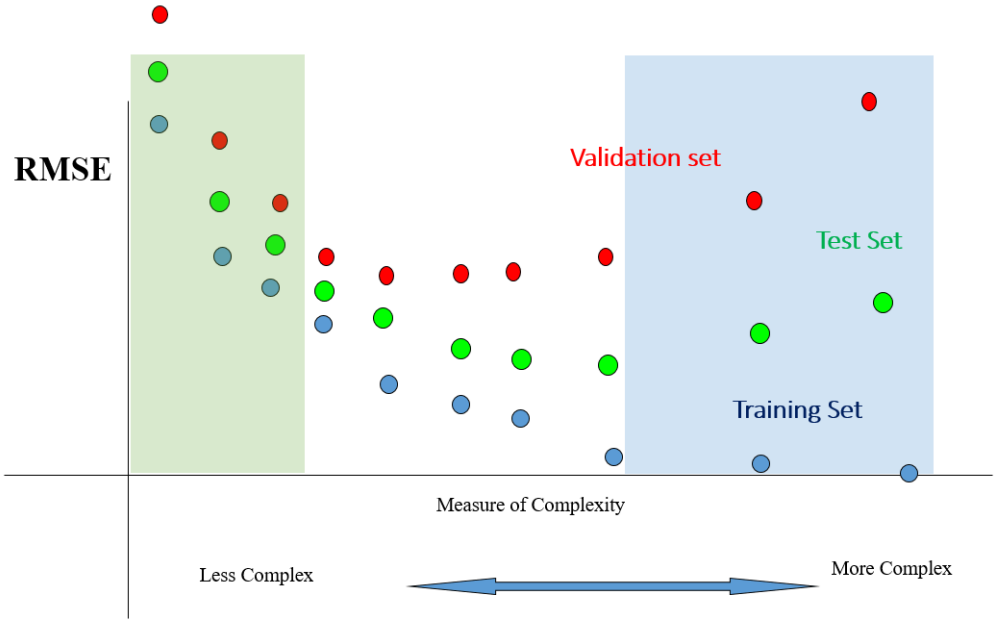
  - For each observation in the validation dataset, compute the prediction error. This is the difference between the actual response and the response predicted by the training set model.
  - SSE to obtain the $SSE_{Validation}$.
  - Denote the number of observations in the validation dataset by $n_{Validation}$.

  In the "Fit Least-Square Crossvalidation" report in *jmp*, which is applied mostly for simulation data analysis in this study, the RMSE for the validation and test dataset is called Root Average Squared Error (RASE).

The models are fit using only the training dataset. In stepwise selection methodology with CV, the $RSquared$ Value from the training dataset and the $RSquared_{Validation}$ Value from the validation dataset are computed in each modeling step. As the number of predictor parameters increases in the model, the higher the complexity of the model is, the RMSE of training dataset decreases monotonically as shown in Figure 5.7a by the blue-dotted curve. The validation dataset $RMSE_{Validation}$ Value, however, has monotonic decrements first in the low model complexity region shown as the green box and then raises as the model complexity goes too high as shown in the blue box region. The green box region shows the models are underfitting. The underfitting models have high bias and low variance estimation.



(a) Cross-Validation with two-split dataset.



(b) Cross-Validation with three-split dataset.

Figure 5.7. RMSE of training dataset (gray dots) and validation dataset (red dots) and test dataset (green dots) vs. training model complexity from low to high. The left green box is the underfitting model while the right blue box is the overfitting model. Adapted from [75].

As more predictors are fit in the model, the model fits better with the sample data until the model complexity reaches the blue box region; the $RMSE_{Validation}$ Value of validation data starts rising and the model is overfitting. This is because the training model starts fitting on the insignificant data that do not represent the general behavior of the sample data. When the model is verified by the independent validation dataset, the goodness of fit starts degrading. Overfitting models have low bias but high variance estimating results.

Because neither underfitting or overfitting models is preferable, proper complexity models are shown within the white region, which the $RMSE_{Validation}$ Value stops decreasing and starts increasing in the CV. The least complex acceptable model in the white region is preferred; i.e., the model that is close to the left boundary of the white region is the well fit model.

NOLH DoE Extension

For the data splits hold out method, a potential drawback of the data subdivision is that a portion of precious sample data are spared for validation purposes rather than for training the meta-model. The data loss from the dataset that has only a small amount of available data may not be affordable. In order to have as many sample data as possible for sufficient CV data subdivision, an NOLH DoE extension study is presented by Ang [76]. By rotation and stacking a two-factor (f=2), five-level (k=5) O_2^5 DoE one time (two stacks, n=2) with row of 0s removed, it generates $k \times n - (n-1)$ DPs and the new set of DPs still keep the same orthogonal property between each factor as the original O_2^5 DoE. The new DPs S_{1stack} is shown in Table 5.2 as the S_1 column DPs stacks with the adjacent S_2 DPs in Table 5.1. With the same operation, S_{2stack} column has $2k - 1 = 10 - 1 = 9$ DPs by rotating the Table 5.1 DoE and then stacking them with the origin DPs for about twice the number of DoE.

In the NOLH extension operation, the five factors, 257 DPs can operate four times rotation, five stacking operations (n=5), which extents the original 257 DP to $257 \times n - (n-1)$ numbers of DPs. The five-factors NOLH DoE has been extended to $257 \times 5 - 4 = 1\,281$ DPs with the same orthogonality between each factor as shown in Figure 5.8, and the range of parameters set up is listed in Table 4.1.

Table 5.1. Two factors S_1, S_2, five levels O_2^5 DoE

S_1	S_2
-2	1
1	2
0	0
2	-1
-1	-2

Table 5.2. An O_2^5 DoE with one stacking operation generates twice as many DPs with one redundant row of 0s removed.

S_{1stack}	S_{2stack}
-2	1
1	2
0	0
2	-1
-1	-2
1	-2
2	1
-1	2
-2	-1

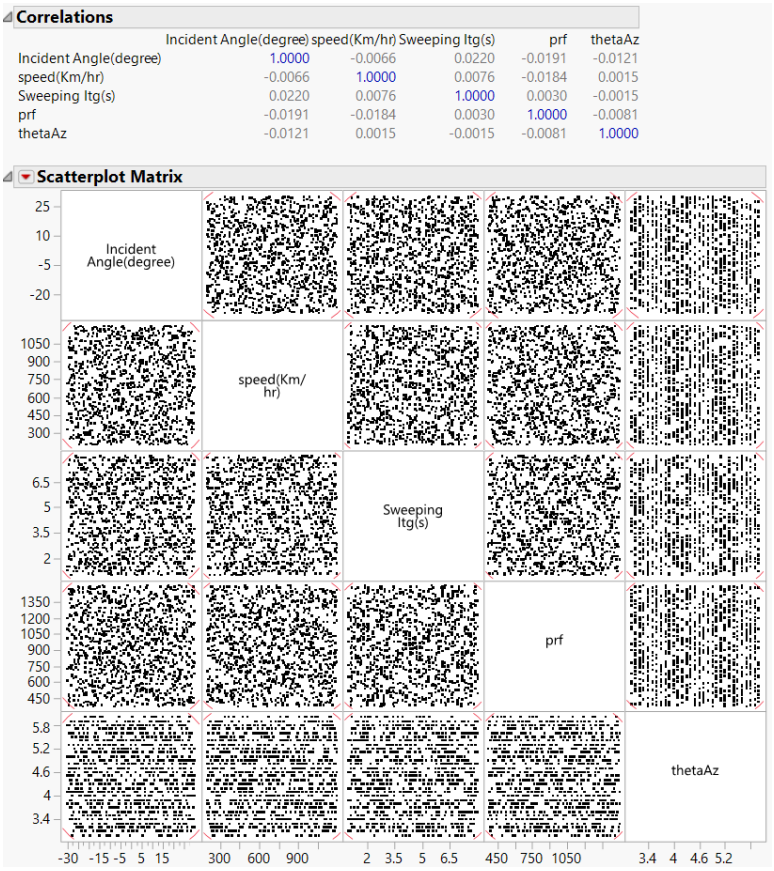


Figure 5.8. Five factors NOLH DoE extended by four-time rotation, five stacking (n=5) operations. The number of DPs is extended from 256 to $256 \times 5 - 4 = 1281$ DPs. The DoE still maintains the same correlation level between each factor.

Three-way Data Splits

Because there is quite a generous number of DPs after DoE extension for this computer base simulation study, the CV holdout method is proper to apply for fit model with plenty of observations. The holdout method that splits the dataset into three groups is introduced. The method splits data for one more test dataset than the two-way data splits method. In both data splits hold out methods, to mitigate the possible misleading results due to an unfortunate split, random subsampling is applied for group selection. The components of the three-way data splits are:

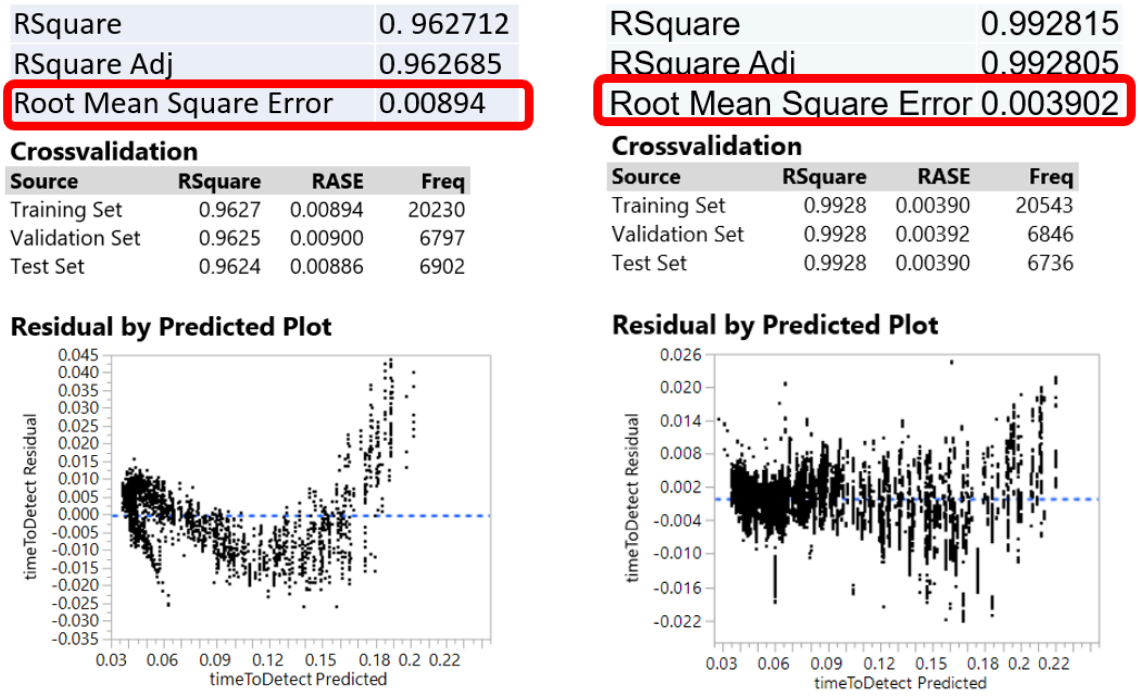
- **Training Set:** It contains the major portion of the dataset used to train for the meta-model. In this case, it is 60 percent of the total available sample data in this study.
- **Validation Set:** There are 20 percent randomly selected data from the total available dataset for the validation set. The validation set is applied for verifying the error rate to the meta-model, which is fit by the training dataset.
- **Test Set:** Another 20 percent randomly selected dataset from the total available dataset are for the test set. The test set is an independent dataset that is only used for testing the RMSE of the selected training model by this independent test dataset. By comparing the RMSE or RASE value between these three data splits, if the validation results are quite consistent between each dataset, it indicates the meta-model should be fit with the proper number of predictor parameters.

Figure 5.9a shows the residual plot of predicted TTD by the model that fit with polynomial and factorial parameters in degree of two by 1,258 DPs generated from extended NOLH DoE. Each DP has been executing 100 replications thus, resulting in 1258×100=125,800 observations for the CV analysis. As the result, the meta-model is fit by randomly selected 20,230 observations while verified by $6,797$ validation datasets for finding the lowest RASE of validation.

The meta-model has quite evenly distributed RASE: 0.00894, 0.00900, 0.00886, are calculated for the Training Set, Validation Set, and Test Set respectively. The result shows no significant sign of overfitting in this model as the model is fit by the predictors with the degree of freedom = 15.

However, the two-degree of interaction of predictors model cannot keep residuals normality

by observing the clear pattern in the residual plot. Thus, the model fit in the order of parameters interaction terms up to four is shown in Figure 5.9. The residual keeps more normality than the two-degree interaction terms model. The MSE has degraded from 0.00894 down to 0.00390. The RASEs are 0.00390, 0.00392, and 0.00390 in the training set, validation set, and test set, respectively. The quite even RASE between the three-split datasets suggests the meta-model is properly fit by the predictors of degree of freedom = 30. The overall prediction expression of this meta-model is in Appendix C.3.



(a) Model residual: 2 degrees of interaction terms. (b) Model residual: 4 degrees of interaction terms.

Figure 5.9. Residual plot by predicted time to detect of the fit models by 3 splits cross-validation with 1,285 DPs. In (a), the model is fit by the training dataset with 2 degrees of interaction parameters. In (b) the model is fit by the training dataset with 4 degrees of interaction parameters.

5.4.2 K-Fold Cross-Validation (CV)

K-Fold CV is a technique that is suited for smaller datasets without discarding holdback data that may reduce the information for fitting the meta-model. In this type of CV, the data are partitioned randomly into k different groups and the following procedure is followed [77]:

1. The whole number of data N are randomly divided into K groups. For $i = 1, 2, ..., K$, the i^{th} group, with the number of $\frac{N}{K}$ data, is treated as a holdback dataset for validation, while the model is fit to the remaining $N - \frac{N}{K}$ observation data for training set. An $RSquared_{Validation}i$ is calculated using i^{th} group as a holdback.
2. After iterating through the K group, the CV RSquared value is set to be the average of the $RSquared_{Validation}i$ for K iterations. It is defined as follows:

$$CV \;\; RSquared = \frac{1}{K} \sum_{i=1}^{K} RSquared_{Validation}i \tag{5.30}$$

3. Apply maximum K-Fold CV RSquared as the stopping rule in stepwise regression analysis. Maximum K-Fold RSquared considers only the model defined by p-value entry (Forward selection) or removal (Backward elimination). It does not consider all possible models.

K-Fold CV is about K times as computationally intensive as holdback CV since it requires K time iterations of computation to obtain a CV RSquared value of a possible model. Therefore, for a complicated model with a large number of predictor factors or when the dataset is large, this method may not be efficient due to the computational complexity. K-Fold CV generally works well for K between 5 to 20 and 10-Fold is a commonly used number in the study.

5.4.3 Cross-Validation CV of Logistic Predicting Model of P_d

The same three splits CV operation is applied to the logistic-regression analysis for predicting P_d by the 125,800 observations which is generated by the extended NOLH DoE. The whole observation data is randomly split into 60 percent for training dataset, 20 percent for validation dataset and 20 percent for the test dataset. The logistic-regression model of P_d is fit by with the polynomial and factorial interaction predictor with degrees of two and four.

Logistic-Regression of P_d with Interaction Predictors to Degree of Two

The measurement of the three splits CV of the logistic model of P_d with two-degree of interaction predictors is shown in Figure 5.11a. The logistic meta-model is fit by the

predictors with degree of freedom=14. The RMSEs of the training set, validation set, and test are 0.1319, 0.1243, and 0.1341 respectively. There is no significant difference in RMSE value between each split dataset. Thus the CV measurement indicates no strong evidence of over fitting.

Confusion Matrix of Probability of Detection P_d

To learn more information about how the logistic meta-model predicts P_d by each DP, a data classification method "Confusion Matrix" of each dataset is introduced. The confusion matrix compares the most alike of predicted detection results by Equation 5.20 with the actual detection result from simulation observations. There are four sets of data classified by the confusion matrix [78]:

1. True positive: These are the cases in which the correctly predicted result is true (the target is detected) while the observation data is also true (the target is detected in actual simulation). The true-positive count is 44699 in Figure 5.10a, which is the result from the training dataset of the logistic model fit with the interaction parameters in degree of two.
2. True negative: These are the data in which the correctly predicted result is false (the target is not detected) while the observation data is also false (no detection in the simulation). The data count is 19641 in Figure 5.10a.
3. False positives: The prediction is true (predicts the target is detected) on the result while the observation data is false (no detection in the simulation). It is also known as a "Type I error." The Type I error rate is calculated by the false-positive count divided by the total true-negative count. The Type I error rate of training dataset is computed as follows and shown in Figure 5.10a:

$$\text{Type I error rate} = 967 \div 45666 = 0.0212$$

4. False negatives: The prediction is false (predicts the target is not detected) while the from observation data is true (detection happens in the simulation). It is also known as a "Type II error." The Type II error rate is calculated by the false-negative count divided by the total true-positive count. The Type II error rate of training dataset is

computed as follows and shown in Figure 5.10a:

$$\text{Type II error rate} = 657 \div 20298 = 0.0324$$

The overall Type I error rates of the logistic meta-model, which is fit with two-degree of interaction predictors, are 0.0212, 0.0171, and 0.0218 from the training set, validation set, and test set, respectively. The false-positive rate is around 2 percent in three randomly selected datasets. The type II error rates are 0.0324, 0.0316, and 0.033 from the training set, validation set, and test set respectively. The false-negative rate of this model is around 3.3 percent, which is shown in Figure 5.10. The type I and II error rates are quite even among the three data splits; thus, it also indicates no significant evidence of over fitting model. This logistic meta-model prediction expression is in Appendix C.4.

Training

	Most Likely Detection		
Detection	0	1	All
0	44699	967	45666
1	657	19641	20298
All	45356	20608	65964

Type I error: 967÷45666=0.0212

Type II error: 657÷20298=0.0324

(a) Training set of logistic P_d

Validation

	Most Likely Detection		
Detection	0	1	All
0	14877	259	15136
1	207	6558	6765
All	15084	6817	21901

Type I error: 259÷15136=0.0171

Type II error: 207÷6558=0.0316

(b) Validation set of logistic P_d

Test

	Most Likely Detection		
Detection	0	1	All
0	15034	335	15369
1	226	6640	6866
All	15260	6975	22235

Type I error: 335÷15369=0.0218

Type II error: 226÷6866=0.033

(c) Test set of logistic P_d

Figure 5.10. The confusion table of logistic Probability of Detection P_d regression model with factorial and polynomial interaction predictors up to a degree of two. The data is divided as follows: 60 percent for training set, 20 percent for validation set, and 20 percent for test set.

Logistic-Regression of P_d with Interaction Predictors to Degree of Four

As the linear-regression analysis of the hybrid model, the two degrees of interaction predictor model has a lack of expressive power to keep the normality of predicted residuals. Therefore, in logistic-regression analysis of P_d, the four-degree of polynomial and factorial interaction predictor parameters are applied in meta-model. This logistic meta-model is fit with predictors with degree of freedom=11. The measurement of CV of this model is shown in Figure 5.11a. The RMSEs are 0.1270, 0.1300, 0.1263 in training set, validation set, and test set, respectively. The difference in RMSEs among the three data splits is more even than the model fit with two-degree of interaction predictors and still no significant difference that is caused by possible overfitting of the model.

Measure	Training	Validation	Test	Definition
Entropy RSquare	0.9074	0.9163	0.9052	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.9503	0.9554	0.9491	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.0571	0.0517	0.0586	∑ -Log(ρ[j])/n
RMSE	0.1319	0.1243	0.1341	√ ∑(y[j]-ρ[j])²/n

(a) Cross-Validation(CV) measurements of logistic model of P_d fit by two-degree of interaction terms.

Measure	Training	Validation	Test	Definition
Entropy RSquare	0.9134	0.9090	0.9132	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.9538	0.9514	0.9536	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.0536	0.0566	0.0535	∑ -Log(ρ[j])/n
RMSE	0.1270	0.1300	0.1263	√ ∑(y[j]-ρ[j])²/n

(b) Cross-Validation(CV) measurements of logistic model of P_d fit by four-degree of interaction terms.

Figure 5.11. The cross-validation (CV) measurements of logistic model of P_d. The data is divided as follows: 60 percent for training set, 20 percent for validation set, and 20 percent for test set. In (a), the model is fit by two-degree of interaction terms with 14 degrees of freedom of predictors. In (b), the model is fit by four-degree of interaction terms with 13 degrees of freedom of predictors.

In confusion matrix analysis, the Type I error rates are 0.0192, 0.0210, and 0.0196 from training set, validation set, and test set, respectively. The false-positive error rate is about 2 percent through all DPs. The Type II error rates are 0.0283, 0.0273, 0.0238 from Training set, Validation set and Test set respectively. The false-negative error rate is around 2.7 percent through all the experiments.

The logistic-regression analysis measurement results show that both the two-degree and four-degree interaction predictors fit model perform quite consistently in RMSE and prediction error rates from the confusion matrix analysis. It can be concluded that both models do not have significant signs of overfitting or underfitting. This four-degree of interaction predictors logistic meta-model prediction expression is provided in Appendix C.5.

Training

Detection	Most Likely Detection 0	1	All
0	44877	877	45754
1	561	19982	20543
All	45438	20859	66297

Type I error: 877÷45754=0.0192

Type II error: 561÷20298=0.0273

(a) Training set of logistic P_d.

Validation

Detection	Most Likely Detection 0	1	All
0	14705	315	15020
1	194	6652	6846
All	14899	6967	21866

Type I error: 315÷15020=0.0210

Type II error: 194÷20298=0.0283

(b) Validation set of logistic P_d.

Test

Detection	Most Likely Detection 0	1	All
0	14903	298	15201
1	160	6576	6736
All	15063	6874	21937

Type I error: 298÷15201=0.0196

Type II error: 160÷6736=0.0238

(c) Test set of logistic P_d.

Figure 5.12. The confusion table of logistic probability of detection P_d regression model with factorial and polynomial interaction predictors up to the degree of four. The data is divided into 60 percent of training set, 20 percent of validation set, and 20 percent of test set.

5.4.4 Sensor Model Implementation by the Meta-models

With the P_d predicted by the logistic meta-model and the TTD predicted by the linear regression meta-model, a two-stage process allows for a complete meta-model structure to predict the behavior of the hybrid model. This complete meta-model construction represents the sensor model in a pure DES architecture while considering high-fidelity physics models of radar parameters. The two-stage prediction sequence by the meta-models is described as follows:

1. **The "if the detection happens" statement predicted by the logistic P_d model:** The probability of detection P_d of the sensor to the target is computed by the logistic-regression model "Probability(Detection)." The "if the detection happens"

statement is determined by the expression as follows:

$$\text{If detection} = \begin{cases} 0, & \text{Probability(Detection)} < \text{RV}\sim \text{Uniform [0,1]} \\ 1, & \text{Probability(Detection)} > \text{RV}\sim \text{Uniform [0,1]} \end{cases}$$

The statement determines whether a detection happens by this set of DoE. The determination is decided by the comparison of the predicted P_d with a Uniform[0,1] RV for a stochastic type of logistic model. If the comparison determines there is no detection by the DoE, the returned result is "no detection."

2. **Predicting TTD when the detection event is true:** If the logistic model predicts the detection is true, then the TTD is predicted by the linear-regression model with the DoE that has been applied for predicting the P_d in the logistic-regression model.

In this research, the logistic meta-model introduced in Appendix C.5 and the linear regression meta-model in Appendix C.3 are applied for constructing this new high-fidelity DES sensor model. The model complexity and execution efficiency of the meta-model is much better than the baseline hybrid sensor model.

5.5 Meta-Model Predicting Results Verification

To verify that the predicted TTD from the fitting meta-model is statistically no different from the baseline hybrid sensor model, two statistical verification examinations are applied on the simulation results from these two models with the same DoE. The examinations verify how significant the difference is in the mean and variance of the simulation results between the fitted meta-model and the hybrid sensor model. The simulation is conducted by a DoE:

- Incident Angle = 1.553°
- Speed = 442.19 km
- Sweeping integer = 6 s
- prf = 1478 Hz

Both the meta-model and the hybrid model perform this DoE with 100 replications. The simulation generates the result with 95 and 80 valid TTDs from the meta-model and the hybrid model respectively.

The two-sample t-test and the unequal-variance test discussed in the previous two sections are the verification method to determine whether the fitting meta-model has good predicting ability to represent the sampling data that is generated by the hybrid model simulation.

However, the validation of the meta-model is verified by examining the mean (first moment) and variance (second moment) of predicted results. Testing the first moment and second moment between predicted results and baseline data can indicate how well the meta-model represents the hybrid model.

5.5.1 Equal-Mean Test: Two-sample t-test

Figure 5.13 shows the result of processing a two-sample t-test on TTD from both the meta-model and the baseline hybrid sensor models with 100 repetitions in one design point. The t-Ratio = -1.07458 with the p-value = Prob > $|t|$ = 0.2853. The result retains the null hypothesis of equal-mean assumption at a 95 percent confidence interval. In other words, the mean of predicted TTDs is statistically identical to the result from the hybrid sensor model.

5.5.2 Unequal-Variance Test: O'Brien Unequal-variance Test

The homogeneity of the variance assumption is one of the critical assumptions underlying most parametric statistical procedures, such as the linear-regression analysis in this study, and it is important to be able to test this assumption. The O'Brien unequal-variance test verifies the variance between the sample variance and population variance by the assumption that [79]:

- The null hypothesis (H_o): this tests that the samples under consideration come from populations with the same variance.
- The alternative hypothesis (H_a): this tests that the populations have different variances than the sample variance.

The O'Brien test is designed to test the homogeneity of the variance assumption for several samples at once and with the versatility for analysis of variance designs, including contrast analysis and sub-designs analysis. The p-value of this unequal-variance test is less than 0.0001. It rejects the null hypothesis of equal-variance assumption. The test indicates the

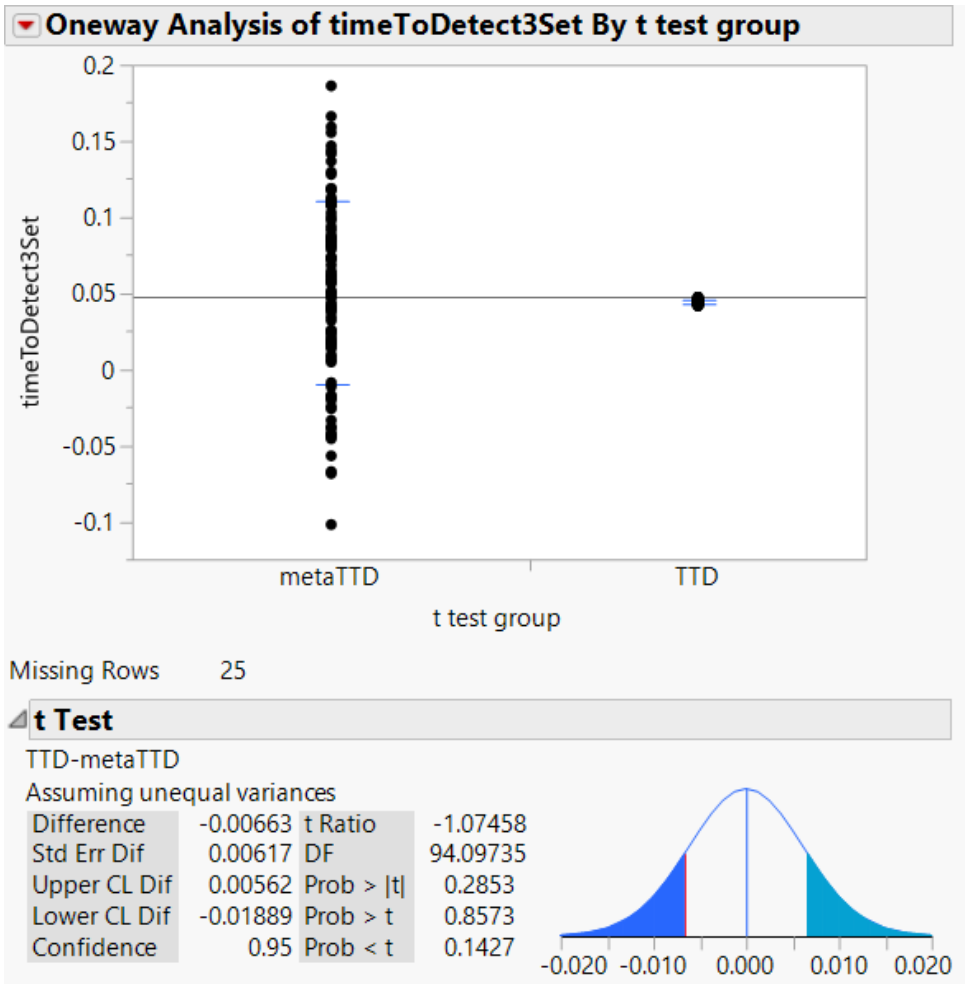
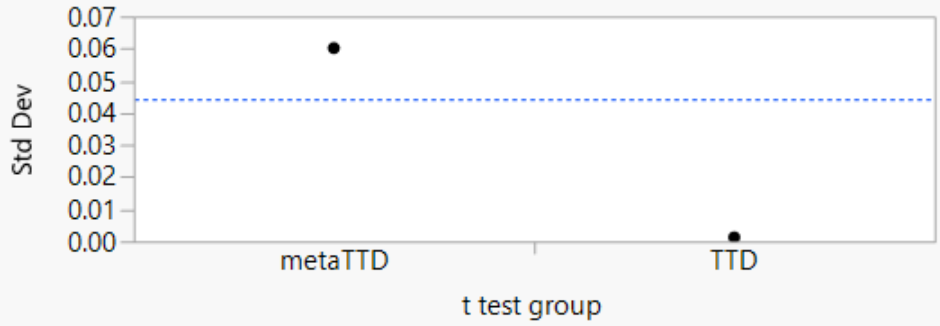


Figure 5.13. t-test of acTTD between the meta-model and the baseline hybrid sensor model.

variance of sample data, which is predicted TTDs in this study, is statistically different from the variance of the population data, which is simulation replication data from the hybrid model. The O'Brien test result is shown in Figure 5.14.

The two-sample t-test has proved that there is no significant evidence to show the mean of the sample data from predicted TTD is different from the population mean of TTD from the simulation data. The unequal-variance O'Brien test shows the sample data from the predicted TTD presents unequal-variance to the population TTD from the simulation data. The fitting meta-model can be concluded as an unbiased prediction response of TTD with unequal prediction variable in comparison to the population simulation data. The overall

Tests that the Variances are Equal



Level	Count	Std Dev	MeanAbsDif to Mean	MeanAbsDif to Median
metaTTD	95	0.0601642	0.0486876	0.0486099
TTD	80	0.0012564	0.0009308	0.0009139

Test	F Ratio	DFNum	DFDen	p-Value
O'Brien[.5]	50.8078	1	173	<.0001*
Brown-Forsythe	143.7439	1	173	<.0001*
Levene	148.8551	1	173	<.0001*
Bartlett	502.8732	1	.	<.0001*
F Test 2-sided	2293.1077	94	79	<.0001*

Figure 5.14. Equal-variance O'Brien test measurement of TTD between the meta-model and the hybrid sensor model.

performance of the meta-models do present the hybrid model characteristics in a certain sense with much more concise structure and with higher executing efficiency.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusions and Recommendations

Essentially, all models are wrong, but some are useful.

–George E. P. Box [80]

6.1 Conclusions

Because the model complexity for integrating high-fidelity physics parameters is high, the simulation efficiency and the model fidelity both have never been satisfied before. A new pure DES architecture sensor model, which predicts only P_d and TTD for the efficient dynamic sensor-model simulation, is successfully constructed in this research. The new sensor model is not only constructed with low model complexity, but also incorporates the high-fidelity physics model of radar parameters. The sensor model detection algorithm considers: 1) range, 2) aspect angle, factors between the sensor and the targets, which are constantly changing in a dynamic sensor-model scenario.

The sensor model constructing architecture is based on a two-stage process. The model construction paradigm in each stage is distinguished and based on different modeling approaches. This chapter presents the conclusions of this high-efficiency and high-fidelity sensor model approach by these two stages:
1) A hybrid architecture of the sensor model that integrates high-fidelity radar equations into a detection algorithm.
2) The concise form of meta-models that represents the hybrid sensor model's characteristics for efficient and simplified sensor model structures through optimized DoE and regression analysis process.

6.1.1 Hybrid Architecture Sensor Model

This work demonstrates a hybrid structure sensor model that successfully integrates high-fidelity RRE with dynamic RCS response from targets. Based on a computationally efficient DES structure with an isolated atemporal RCS model, the hybrid model not only introduced an angle-varying RCS factor into the dynamic sensor-target scenario, but also kept the computational complexity of a large complex scenario down to a tractable level.

A typical desktop PC is capable of executing 250,500 DoE scenario repetitions in a matter of minutes. In extending the NOLH DoE for generating more observation data for more flexible and detailed study of the proceeding regression analysis process, the number of DPs has risen five times, up to 5× 250,500 = 1,252,500. With such a tremendous number of simulation replications, the execution latency is still controlled to under a minute or two. This is because the three categories classification structure classifies the data models that contain different time mechanisms in simulation into associated groups and integrates each group accordingly into a model. The basic DES-based sensor framework maintains the simulation accuracy by scheduling and executing the simulation precisely on the event transition state. This brings no ambiguity to the simulation event scheduling time for emulating the TS antenna scanning mechanism. The DES sensor framework also optimizes the simulation execution efficiency by ruling out the events without state transition during the simulation.

Furthermore, the atemporal model is a newly proposed modeling time mechanism classification in this research. Such time-independent factors, which can possibly cause tremendous execution latency during the simulation, have to be processed separately and a priori. In this research, the RCS of the targets varies according to the aspect angle to the sensor without a time factor inside. However, the tremendous computational complexity of generating RCS makes it impossible to be incorporated into nearly-a-real-time or prompt-time-varying simulation scenario; doing so would require vast computational power to shorten the computational latency, such as being computed instantly by powerful clusters. Such cluster power is not always an available resource for simulation designers and users. One difficulty when using the cluster for simulations is not only the price of the hardware investment but also accessibility to the cluster resource during the simulation. Without stable and high-performance networking for prompt data exchange between the simulation administration terminal and the clusters, any instead previously cluster supported architecture cannot be performed efficiently. When the target RCS is computed offline, this time independent factor can be integrated into the DES framework without noticeable responding latency and the simulation executive complexity can be kept down to a level executable by PC.

The contribution and goal of this hybrid sensor structure is to provide an architecture that allows the highly detailed physics parameters to be integrated and worked coherently and

efficiently into a high-fidelity simulation scenario [81].

6.1.2 DoE and Regression Analysis

Even with these benefits of a DES-based structure for the hybrid model, the embedded TS mechanism for emulating the antenna sweeping period is still a key latency for a comprehensive model. Additionally, the model also needs to include a maximum detectable range table for the target. These factors still make the model bulky in some sense. This research utilizes efficient NOLH DoEs that provide a well-filled space of parameters with significantly fewer design points compared with full factorial design. The 257 DPs DoE by NOLH design with 100 replications of each DP allows the behavior of the hybrid sensor model to be revealed comprehensively in the range of interest for the parameters in the hybrid model.

Finally, a linear regression meta-model is fit from the experimental data executed by the DoEs for predicting the TTD of the sensor to the target with the predictor parameter in the range of DoE. A logistic regression analysis is applied for predicting the P_d of the sensor to the target in a specific DoE. The linear regression fit model showed some heteroscedasticity in predicted TTD when the model is fit by low degree of polynomial and factorial interaction predictor parameters. With the four degree of interaction predictors, the meta-model has good normalized distribution in the predicted TTD residual. Multiple model selection and verification methodologies are applied for attempting to find the proper predictors to fit the meta-model. The optimum model selection processes that have been considered in the study are:

1. **Fit the models with interaction terms up to degree of four:** Due to the obvious heteroscedasticity, in predicted residual when the models are fit with two degree of interaction terms of predictors, the four degree predictors terms mitigate substantial heteroscedasticity and the meta-model has good normalized distribution in the predicted TTD residual.
2. **Dividing data into three datasets for CV:** For insuring the meta-models are not over fitting, the CV process is crucial to verify the models are formed by the proper predictors. Thanks to the substantial numbers of DoE that can be generated by the extended NOLH design that is introduced in section 5.4.1, each dataset has sufficient

observations to verify the model performance. The data are randomly divided into three sets proportionately as follows:

- Training dataset (60 percent)
- Validation dataset (20 percent)
- Test dataset (20 percent)

3. **Stepwise selection :** Forming the meta-models by the forwarding "stepwise" model selection process with the stopping rule of "max validation RSquared." This not only allows users to choose the models from different steps but also allows the validation dataset to verify the fitness of the meta-model that is fit by the training dataset.
4. **Backward elimination:** Once the model is chosen, the backward elimination method is applied for eliminating the insignificant predictors. The eliminated predictors are chosen one by one by the p-value threshold=0.05.

By these model selection and verification processes the meta-model for predicting TTD has RSquared value = 0.993 and the logistic regression model for predicting P_d to the target has RSquared value = 0.94. The error rate shown by this RSquared value is quite low. This may cause the attention whether the models are overfitting. There are multiple model selection and verification methodologies applied for fitting the models with the proper predictors. There is no quick and simple answer about how to fit a meta-model correctly. However, after the application of multiple model selection methodologies and the strict CV by two abundant, independent validation datasets and test datasets, the degree of possible over fitting models can be mitigated.

The meta-models do present the hybrid model characteristics in a certain sense with much more concise structure and with higher executing efficiency.

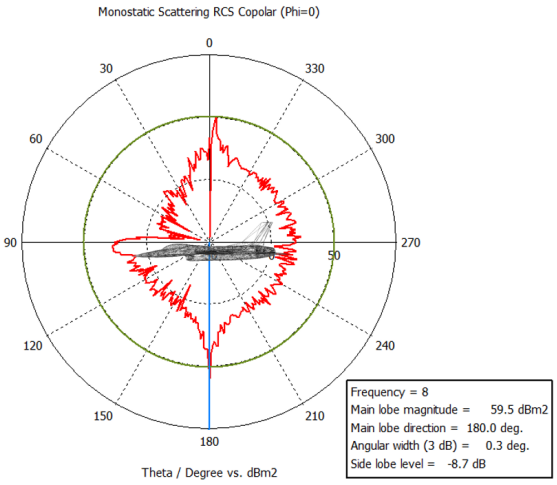
6.2 Recommendations for Future Work

In this research, there is a two-stage approach to complete the model constructing process. The process of each stage is distinguished and unique. The suggestions for extending future work are also described into the two major fields.

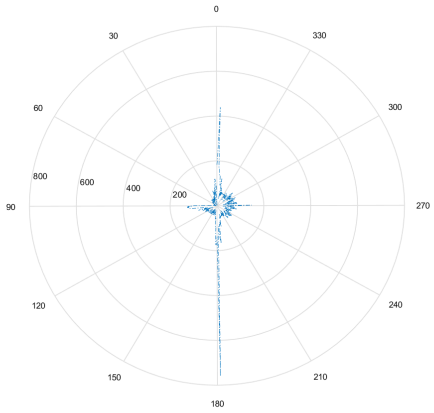
6.2.1 Hybrid Architecture Sensor Model

1. **Extending the 2D planar Hybrid Sensor into 3D RCS Resolution Sensor Model**
   The hybrid sensor model that has been studied in this research is a two-dimensional (2D) planar resolution sensor model. It is a good approach for a 2D search radar sensor model in the far-field that makes no significant elevation angle change as the targets close in on the sensor. Nevertheless, the RCS has agile variation along with the change of the object geometry which is explained intensively in section 2.3.1. As the target moves close to the sensor in the near-field, the difference of elevation angle to the target between each antenna scanning period becomes more significant. In Figure 6.1a displays the RCS variation of the F-16 model in elevation angle. Applying the RCS in different elevation angle into range equation, the Rmax of the target from top ($\theta = 0°$) to bottom ($\theta = 180°$) is shown in Figure 6.1b and the big RCS value happens at the top and bottom of the target. Therefore, by introducing the elevation angle factor into the target RCS, the sensor-target simulation scenario will not only be extended from 2D planar model into 3D simulation resolution scenario, but also consider possible significant elevation factors into the sensor detection algorithm. In Figure 6.1c shows the 2D planar sensor with RCS of the target in $x - y$ plane. Thanks to the flexible and efficient hybrid sensor model architecture, more radar parameters can be integrated into this framework in order to make the sensor detection algorithm more thorough. The hybrid sensor model need to be modified as follows:

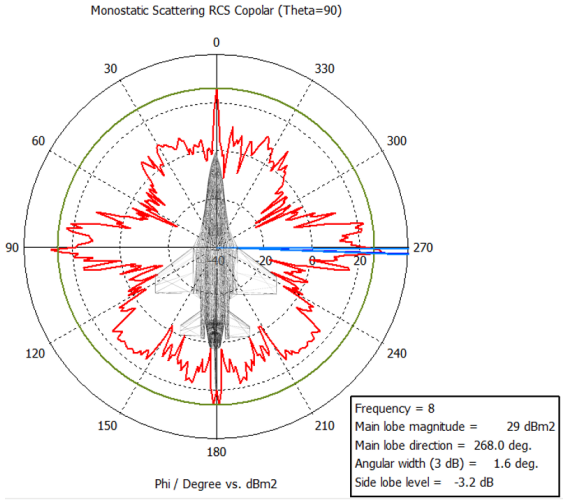
   (a) **Generating 3D Resolution RCS of Targets**:
   RCS is one of the significant factors that can affect the radar model detection ability, but the RCS of the target, which is classified as atemporal model, has high computational latency for generating high resolution results. If not computed in advance, the computational latency can be more severe in 3D resolution. For large aircraft models constructed by high amount of polygons and executed in real size, a PC can get easily overwhelmed due to the massive execution complexity. The partial section computation arrangement can be a possible solution to share the work load on a PC.



(a) RCS of F-16 in elevation angle



(b) Rmax of F-16 in elevation angle



(c) RCS of F-16 in azimuth angle

Figure 6.1. The RCS of F-16 Falcon fighter model which is simulated by CST Studio software with signal frequency = 8 GHz. In (a), the RCS of F-16 sweeping by the elevation angle with $\phi = 0°$. In (c), the RCS of F-16 sweeping by the azimuth angle with $\theta = 90°$. In (b), the maximum detection range to the sensor is computed by the target RCS in part(a).

(b) **Applying Double-Precision 3D Geometric Point in Simulation:**
The geometric point in the 2D model is defined by Point2D format in Java. In the 3D resolution scenario, all the entities utilizing the geometric point need to be updated to double-precision Point3D format in java.

2. **Constructing DES Model by Visual Simkit (Viskit)**
Viskit is a visual programming tool to create, run and analyze Simkit models. Even though the ground truth concept of implementing DES by Simkit or Viskit is the same, the difference is the Viskit constructing models by building entity blocks and assemble the entities as designing event diagrams. The goal for implementing DES by Viskit is to assemble the simulation by dragging and dropping the entity blocks to match the design event graph. This tool can help to reduce implementing syntax error and the simulation event graphs are clear during the assembling [82].
3. **Model visualization using composable X3D**
Further modeling and visualization using composable, archival X3D forgoes can support further understanding and illustration of simple results from complex details.

6.2.2 DoE and Regression Analysis

1. **Applying Advanced Regression Analysis Methodologies**
The prediction result from meta-model for TTD shows the unequal-variance in comparison of the simulation result from the hybrid model. There is always some prediction error between meta-models and hybrid models, even though there are many model selection and verification methodologies have been applied for attempting to get the optimum meta-models in the study. Nevertheless, there is no one methodology of how to choose the best meta-model, especially when dealing with the high complexity simulation scenarios such as the physic-based sensor model. Therefore, further data analysis and regression methodologies can be applied in future work to form a better and more accurate understandings of meta-models of interest.
2. **Developing Multiple Entities Simulation Scenarios**
There are eight different aircraft models have been built by the hybrid structure sensor model based on their RCS in 2D resolution. In this study, only the F-16 Falcon fighter model is used as the baseline model to study the DoE and the regression analysis methodologies. The meta-model expression of the F-16 Falcon fighter is listed in the Appendix C. The other seven aircraft hybrid models can also be simplified as

the meta-models by following the summarized regression analysis methodologies, which are introduced in the conclusion. With result of the eight meta-models with the different type of aircraft, a high-fidelity and dynamic sensor-target simulation scenario with multiple entities can be conducted and be executed efficiently. Following the hybrid-sensor model architecture and the regression analysis methodologies, which has been introduced in the study, new target entities or other physically based models can be introduced into the simulation scenario as further research indicates.

3. **Developing Tactical Flight Waypoints Simulation Scenarios**
   The maximum detection range of the sensor to the targets is highly depending on the aspect angle from the sensor the targets. Figure 3.12 shows the detection range is far at front, back and side of aircraft but much shorter at the diagonal angle of the plan. There are two interesting scenarios can be conduct based on these high-fidelity models for studying tactical waypoints of the aircraft or the radar site arrangements of the sensor.
   (a) **Minimized the detection range from the sensor:** Since the aircraft has short detection range to the radar at diagonal angle, it is possible to find the flight waypoints that get the most small CPA to the sensor such as zig-zag rout that keeps the aircraft facing the sensor with the low RCS side for the longest period of time.
   (b) **Optimum radar site arrangement:** There are more blind angles for a single-site radar to detect the target during a tactical flight. A multiple radar sites arrangement is a possible solution for uncovering the blind angles of a single-site radar. Based on the highly detailed and efficient sensor target models architecture presented in this research, the performance of a multiple radar site arrangement can be examined efficiently with high-fidelity.

APPENDIX A:
X3D Objects Animated by Simulink

Matlab is a powerful tool to compute high-fidelity engineering model and plot the result in figures. Simulink implements Matlab .m source code into block diagrams and flow charts to execute the simulation. This project demonstrates how physics equations implemented in Simulink can animate X3D or VRML models, along with the methods to convert Matlab .fig format into an X3D object so the Matlab 3D object can be applied into Web-based animations. The complete project example is available at:

X3D for Advanced Modeling Examples Archive in
http://x3dgraphics.com/examples/X3dForAdvancedModeling/Matlab
The conversion and operation procedure is described as follows:

A.1 Matlab.m Plot to X3D Objects Conversion

Exporting Matlab .fig format into .wrl object
The Matlab function that exports *.fig to *.wrl by vrml.m function is created by Jan Danek of Humusoft [83]. The function resource is
http://www.mathworks.com/matlabcentral/fileexchange/48242-vrml-h–filename–opts-/content/vrml.m

The exporting procedure is:

1. Generating a 3D plot *.fig in Matlab.
   In this example, a phase array antenna model created by Dr. David Jenn is applied [23]
   (a) Copying "TwoD_array_dips_gp_xz_scan_1plot.m" in Matlab current execution path.
   (b) In command line, type " TwoD_array_dips_gp_xz_scan_1plot" and then the following plot will be generated. The parametric equations for each axis are shown in the labels of Figure 2.2a.
2. Copying "vrml.m" in current execution path.In Matlab command line type: vrml(

gcf, ‘outputFileName.wrl’) and then the exporting acvrml file will be put in Matlab current path. The converted .vrml object is shown in Figure 2.2c.

Exporting Matlab .fig format to X3D and .xhtml

The exporting function is created by Dirk-Jan Kroon [84] and the function resource is: http://www.mathworks.com/matlabcentral/fileexchange/32207-matlab-3d-figure-to-3d--x-html

The exporting procedure is:

1. Copying “figure2xhtml.m” in Matlab current execution path.
2. Generating a Matlab .m figure and then typing in command line : figure2xhtml (‘outputFileName’)
3. By default, there will be two output files generated at current path.
   (a) Exporting X3D object. The “ outputFileName.x3d” object is shown in Figure 2.2b.
   (b) Exporting XHTML object. The “ outputFileName.xhtml” object is shown in Figure 2.2d.

A.2 X3D Object Animated by Simulink

The project remodels the Bouncing Ball example in Matlab VRtool tutorial [85].

The block diagram of the Simulink animation is showing in Figure 2.3 and the parameters are set up as follows:

1. Initializing Simulink diagrams.
   (a) In “Constant” block, the gravity force= 9.8(m/s) is set.
   (b) In “Position” block, set up initial position (Z-axis) = 20(m).
   (c) In “velocity” block, set up initial velocity=0.
   (d) In “fcn” MATLAB Function block, it calculates the force “f”, and X,Y-axis scaling “a”, and Z-axis translation “b” variation during simulation by the input variables position “Z” and velocity “v”.
2. Double clicking "VR Sink" the simulation scene will appear as the right plot in Figure 2.3. In the simulation scene, choose "Simulation/Block", a parameter setting window

will appear as Figure A.1.

(a) In Simulation/Block parameters, the 3D model in *.wrl format can be chosen.

(b) In VRML Tree, check the interested transformation variables to render during the simulation. In this example, "scale and “translation” are chosen.

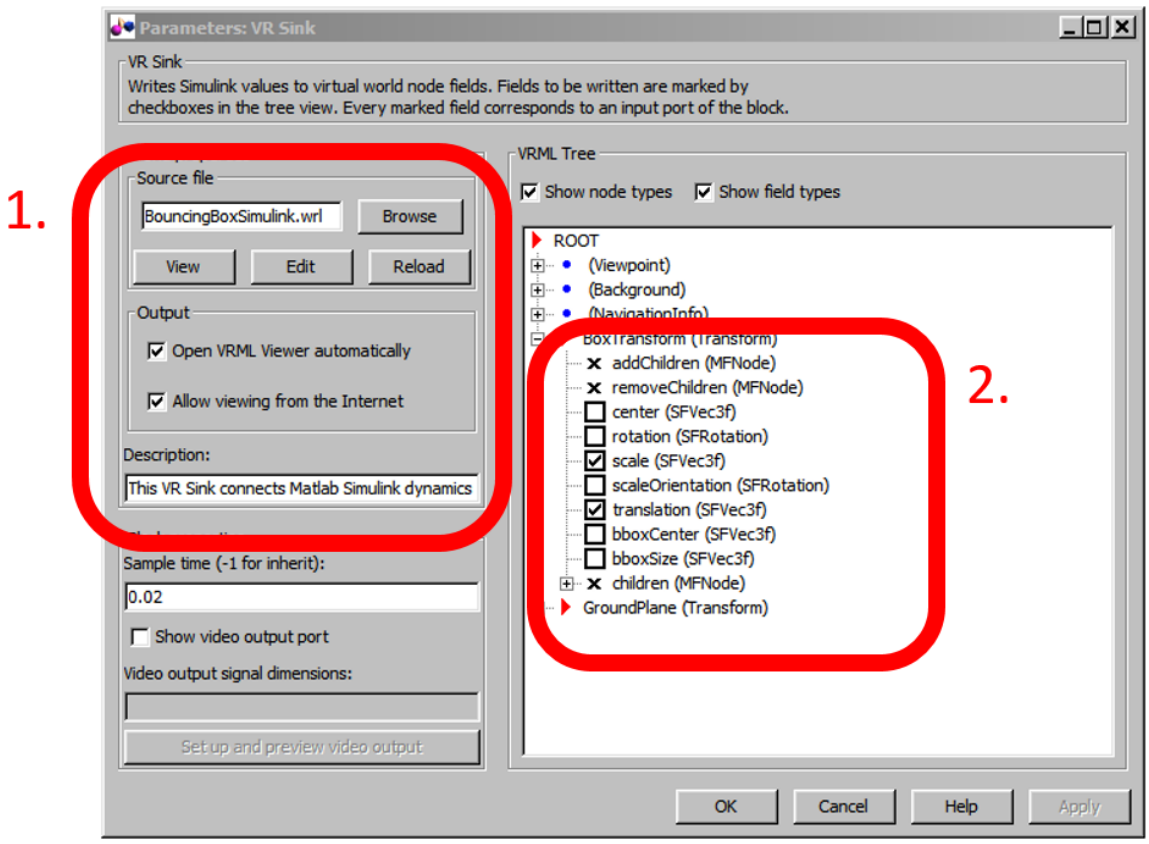


Figure A.1. VR sink software: Parameters settings block-diagram panel.

3. Double click "VR Signal Expander" icon as in Figure A.2, a function parameter set up block is shown as Figure A.2

(a) Output width = 3 indicates there are three parameters of output(X, Y, Z in this case).

(b) The “Position” variable only goes into Z-axis by setting “Output signal indices = [3]”.

During the simulation, there are three scopes monitoring the box movement parameters:

1. The Position Scope:

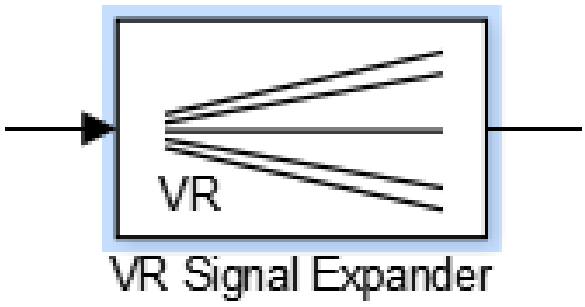


Figure A.2. VR signal expander software: Simulink icon.

It shows the position change (z-axis) of box and is shown in the bottom right plot of Figure 2.3. Since there is the damping function concerned in “fcn”, there is the attenuation of Z-axis movement.

2. The Velocity Scope:
   It shows the velocity change (z-axis) of box and is shown in the bottom plot of Figure 2.3. Likewise, the attenuation of velocity can be observed.
3. The Force Scope:
   It shows the resilient force of box in each bounce and is shown in the bottom right plot of Figure 2.3.

The animation can also be shown in the local host web browser in *.HTML format at http://localhost:8123 and the viewpoints can be switched by “Page-up” and “Page-down”.

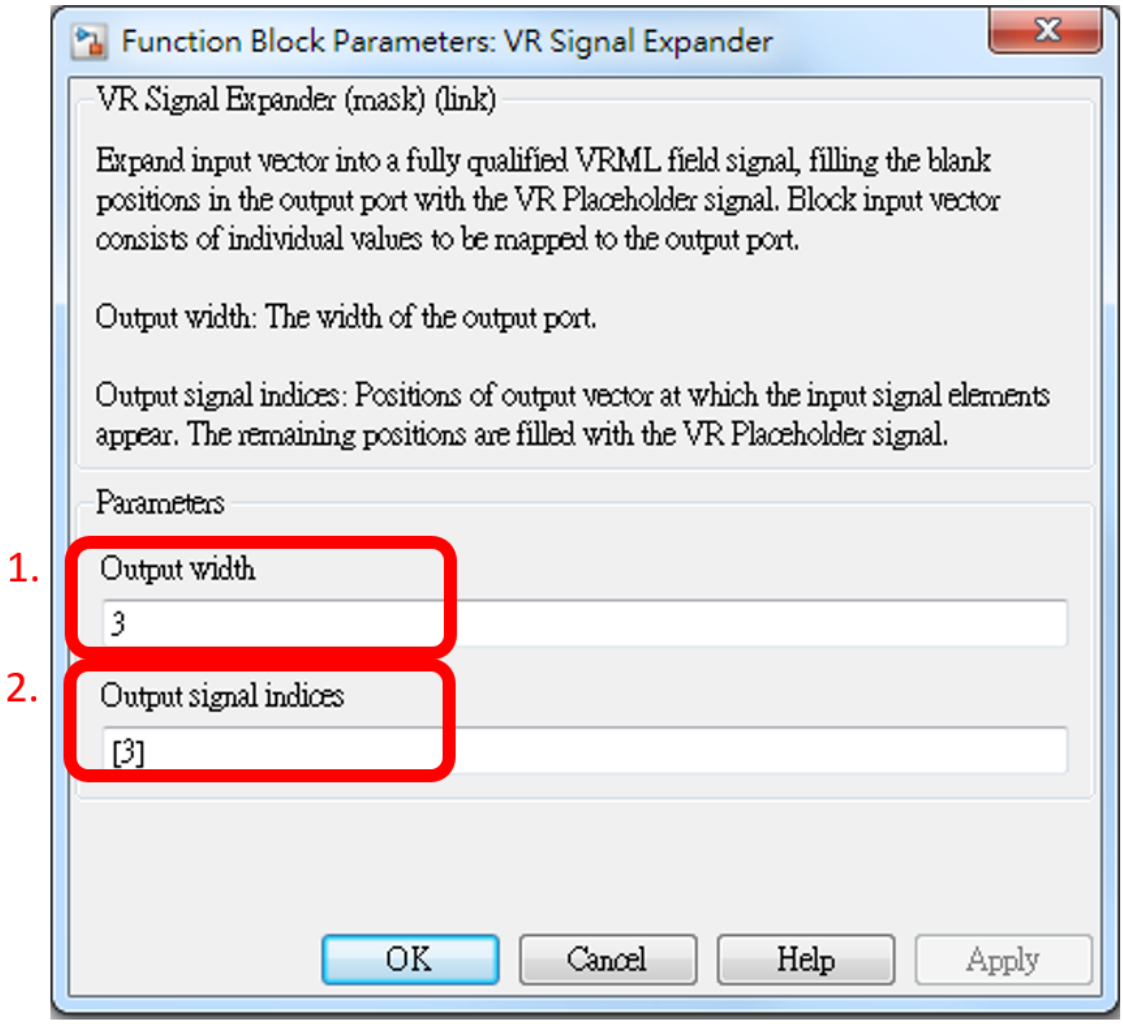


Figure A.3. VR signal expander software: Parameters Setting block-diagram panel.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B:
Hybrid Sensor Model Simulation Operator Manual

The simulations are constructed on Java platform by the Simkit library. The introduction of the Simkit library can be found in the website: https://eos.nps.edu/Simkit/ [57].

The hybrid model simulation packages are available at: The Savage model archive. https://savage.nps.edu/Savage/

B.1 Hybrid 2D Planar Sensor Simulation User Manual

B.1.1 Batch Run Simulation

The hybrid sensor architecture has shown it can integrate the high-fidelity physics-based radar parameters efficiently into a simulation scenario. With a flexible architecture, the simulation can adapt to different aircraft objects and different DoEs for extending the simulation scenarios. The simulation software environment setups are listed as follows:

- **Package:** "RmaxRCSVaring.run"
- **File:** "RCS2DSensorBatch.java." This java main file utilizes the entity files in "RmaxRCSVaring.smd" package and the utility files in "RmaxRCSVaring.util."
- **Variable setting:** The instance variables for initializing the simulation are listed as follows:
    1. **maxRange:** This is the maximum detection range of the sensor. It is denoted "Ru" in the dissertation. It is in the unit of kilometers.
    2. **moverInitialDistanceFromSensor:** The initial distance between the target to the sensor. It is in the unit of kilometers.
    3. **isCoherent:** A Boolean variable for determining if the Swerling model is applied for representing the fluctuation error effect. When the variable is "true", the Swerling model is enabled and the sensor model is stochastic.
    4. **maxEndMoveNumber:** This is the the number of replications per design point.
- **Input file setting:**
    1. The Rmax of the target is imported as the file name "RmaxRCSVaringF16_8G.csv"

by the following code:
String RCSInputFileName = "RmaxRCSVaringF16_8G.csv";

2. The DoE of the batch simulation is imported as the file name "input_29_intg.csv" in the following code: String NOHLFile = filePath + "input_29_intg.csv";

- **Output file setting:** The simulation output is saved as the file name " output.csv" in the following code:
  File outputFile = new File(filePath, "output.csv");
  The output file concatenates the input DoE parameters with the output "timeToDetect" results in each replication. "NoDetection" records 1 if no detection event happens in the scenario and 0 is recorded otherwise.
- **Meta-model implication:** The logistic model expression is implemented in the variable "lin" and P_d prediction is computed by the variable "probDetection." The meta-model of predicting TTD is implemented in the variable "metaTimeToDetect" when the predicted P_d is true.

B.1.2 The Hybrid Sensor Model Simulation Visual Animation Demonstration:

- **Package:** "RmaxRCS2DSensor"
- **File:** "RCS2DSensorDemo.java." This java main file utilizes the entity files in "RmaxRCS2DSensor.smd" and "mv3302.smd" package and the utility files in "RmaxRCS2DSensor.util" package .
- **Variable setting:**
  1. **maxRange:** Set up the maximum detection range of the sensor.
  2. **sensorPingPeriod:** Set up the sensor antenna sweeping integer.
  3. **pathList:** Set up the waypoints. The waypoints follow the java Point2D format.
  4. **Speed:** Set up for the target moving speed.

  By executing this file, a simulation sandbox will appear. Clicking the run button causes the simulation to be performed with the event log displayed on the right side of the dialog window.

B.2 Target Model Radar Cross Section (RCS) and the Maximum Detection Range to the Sensor Model

There are the lists of eight aircraft and missile models, which have been applied in the hybrid sensor architecture in this study. The RCS of these 3D models have been calculated by the CST Studio in the variable of azimuth angle. The 3D models are aligned along x-axis with front side facing positive x-axis direction. The models are set in $x - y$ plane with top side facing to positive z-axis. The RCS is computed around the model on $x - y$ plane which is $\theta = 90^\circ$ while ϕ sweeping from 0° to 360° in spherical coordinate system with 0.5° increment.

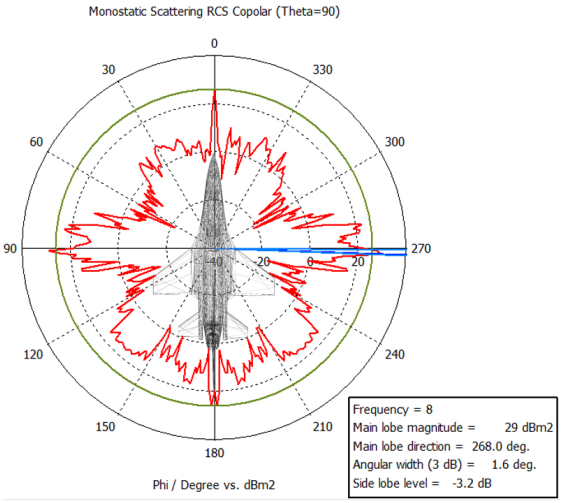
The Rmax of the model to the sensor is calculated by the range equation introduced in Chapter 2.3.1 with the RCS of the target. The polar plot of the Rmax displace the detection range of the target by the viewed angle of the sensor to the targets. The Rmax is in the unit of km.

All models are full size as the real aircraft in CST Studio RCS simulation. In this chapter, the objects are shown in RCS contour plot merging with the object projected figure which matches to the computational configuration. Also the contour Rmax plots are shown and the data are applied in the sensor model for detection determination in simulation. The four different viewpoint plots of the 3D objects are displayed for better understanding of the objects geometry.

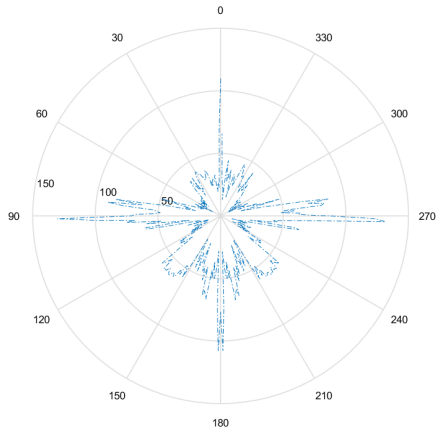
The 3D aircraft models included in this research are purchased from:3Dcadbrower.com.

B.3 Conventional Military Aircraft

B.3.1 F-16 Falcon Fighter RCS and Rmax



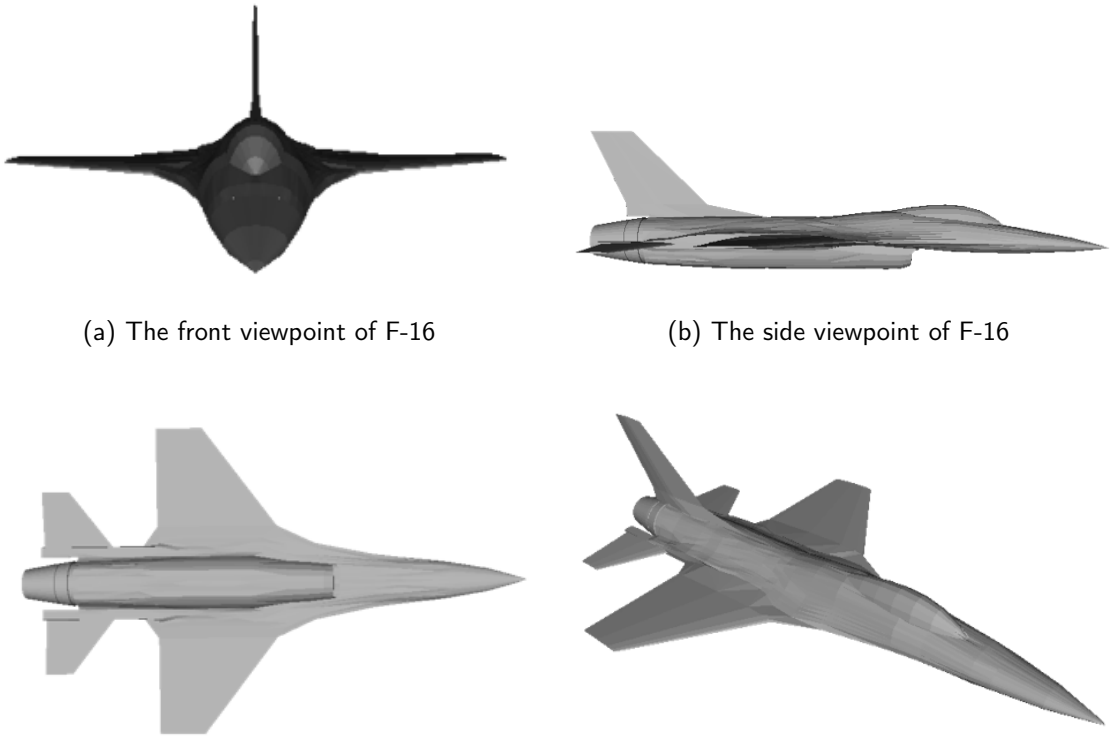
(a) The RCS of F-16



(b) The Rmax of F-16

Figure B.1. The polar plots of a F-16 Falcon fighter in azimuth angle ($\theta = 90°$). In (a), the RCS of the object computed by CST studio with the signal frequency = 8GHz. In (b), the Rmax of the sensor to detect the target and calculated by the range equation with the RCS.

B.3.2 F-16 Falcon Fighter Model in Different Viewpoints



(a) The front viewpoint of F-16

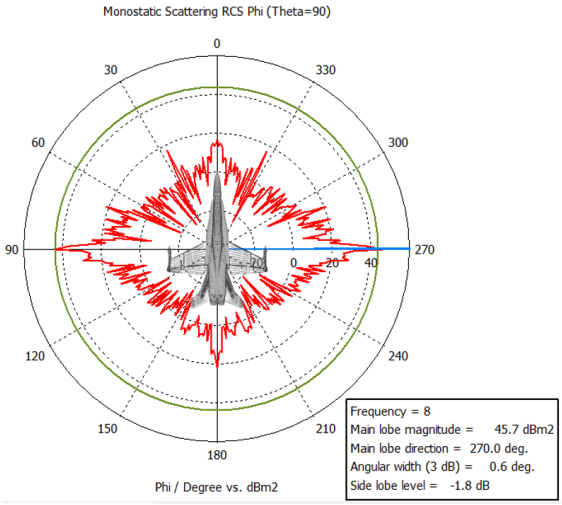
(b) The side viewpoint of F-16

(c) The bottom viewpoint of F-16

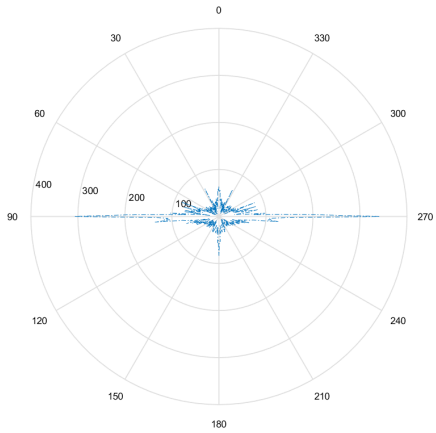
(d) The 45° viewpoint of F-16

Figure B.2. Four different viewpoints of the F-16 Falcon fighter model.

B.3.3 F-18 Hornet Fighter RCS and Rmax



(a) The RCS of F-18



(b) The Rmax of The RCS of F-18

Figure B.3. The polar plots of a F-18 Hornet fighter in azimuth angle (θ = 90°). In (a), the RCS of the object computed by CST studio with the signal frequency = 8GHz. In (b), the Rmax of the sensor to detect the target and calculated by the range equation with the RCS.

B.3.4 F-18 Hornet Fighter Model in Different Viewpoints

(a) The front viewpoint of F-18

(b) The side viewpoint of F-18

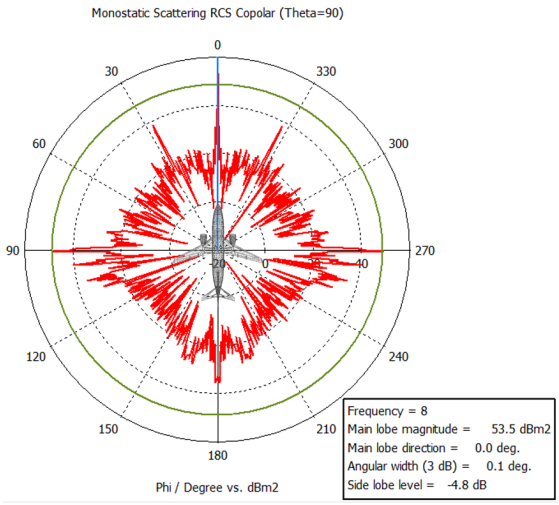
(c) The bottom viewpoint of F-18

(d) The 45° viewpoint of F-18

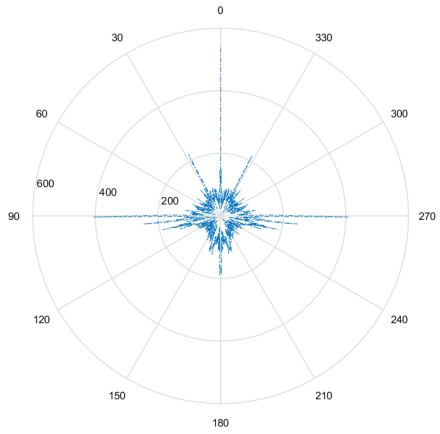
Figure B.4. Four different viewpoints of the F-18 Hornet fighter model.

B.4 Commercial Aircraft

B.4.1 B-737 400 RCS and Rmax



(a) The RCS of B-737 400

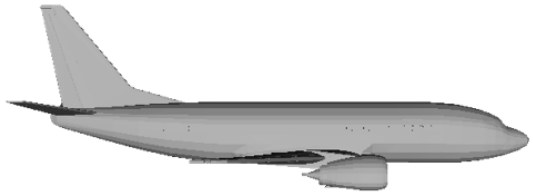


(b) The Rmax of The RCS of B-737 400

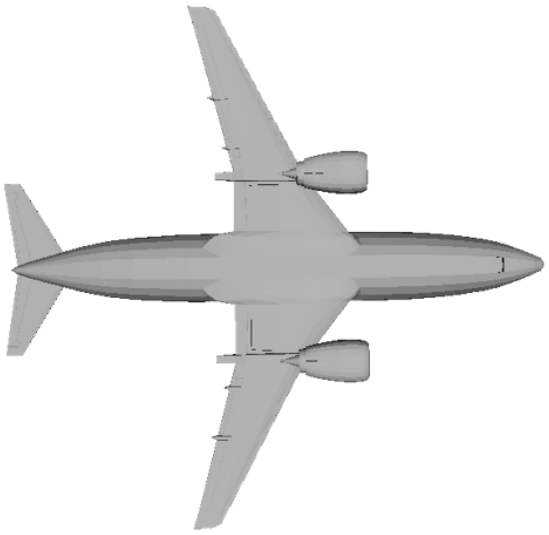
Figure B.5. The polar plots of a B-737 400 commercial aircraft in azimuth angle (θ = 90°). In (a), the RCS of the object computed by CST studio with the signal frequency = 8GHz. In (b), the Rmax of the sensor to detect the target and calculated by the range equation with the RCS.

B.4.2 B-737 400 Model in Different Viewpoints

(a) The front viewpoint of B-737 400



(b) The side viewpoint of B-737 400

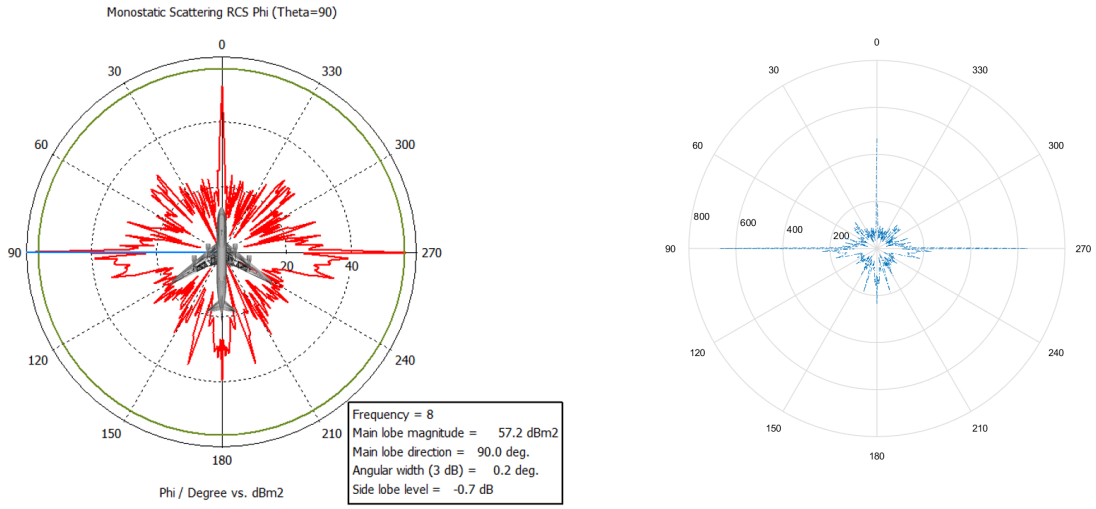


(c) The bottom viewpoint of B-737 400

(d) The 45° viewpoint of B-737 400

Figure B.6. Four different viewpoints of the B-737 400 commercial flight model.

B.4.3 B-747 400 RCS and Rmax

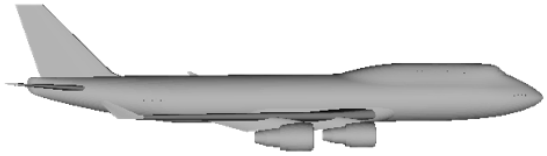


(a) The RCS of B-747 400 (b) The Rmax of The RCS of B-747 400

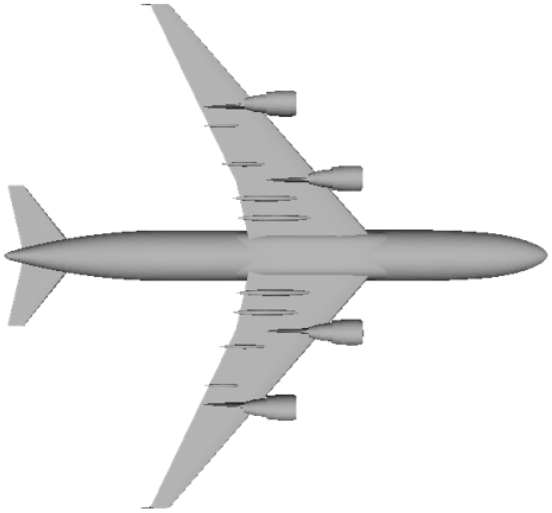
Figure B.7. The polar plots of a B-747 400 commercial aircraft in azimuth angle ($\theta = 90°$). In (a), the RCS of the object computed by CST studio with the signal frequency = 8GHz. In (b), the Rmax of the sensor to detect the target and calculated by the range equation with the RCS.

B.4.4 B-747 400 Model in Different Viewpoints

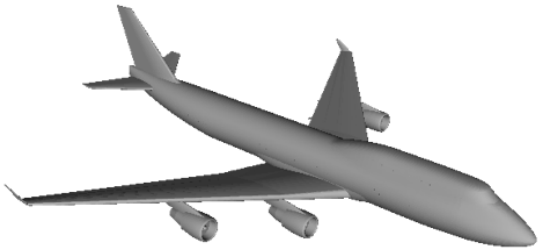
(a) The front viewpoint of B-747 400



(b) The side viewpoint of B-747 400



(c) The bottom viewpoint of B-747 400

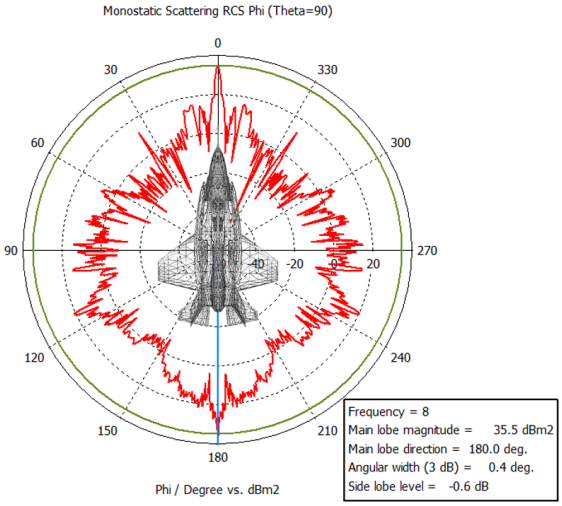


(d) The 45° viewpoint of B-747 400

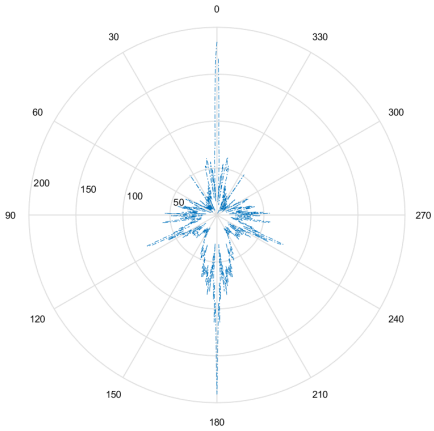
Figure B.8. Four different viewpoints of the B-747 400 commercial flight model.

B.5 Stealth Aircraft

B.5.1 F-35 Lightning Fighter RCS and Rmax



(a) The RCS of F-35



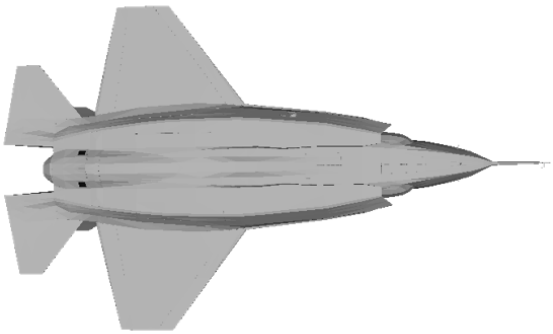
(b) The Rmax of The RCS of F-35

Figure B.9. The polar plots of a F-35 Lightning fighter in azimuth angle (θ = 90°). In (a), the RCS of the object computed by CST studio with the signal frequency = 8GHz. In (b), the Rmax of the sensor to detect the target and calculated by the range equation with the RCS.

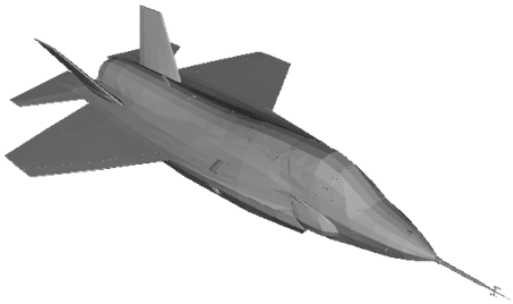
B.5.2 F-35 Lightning Fighter Model in Different Viewpoints

(a) The front viewpoint of F-35

(b) The side viewpoint of F-35



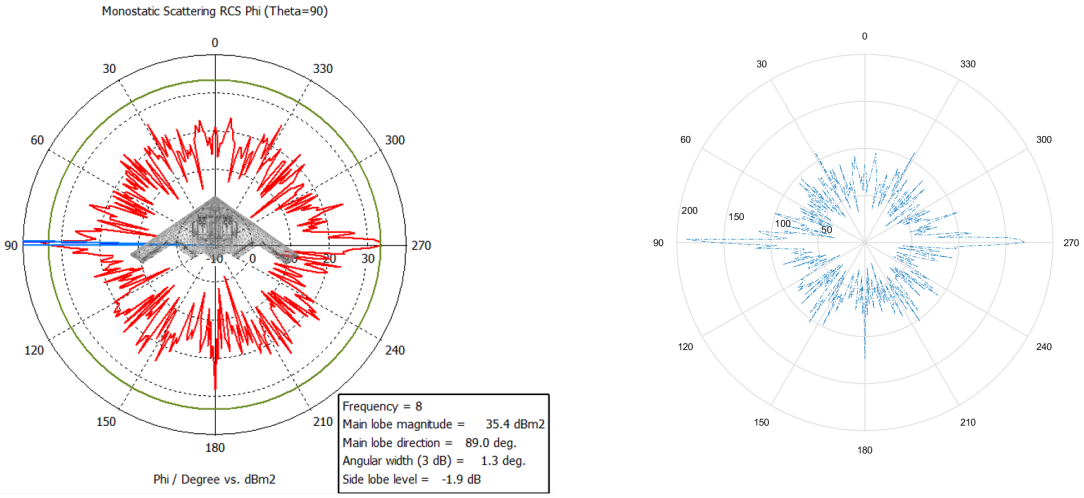
(c) The bottom viewpoint of F-35



(d) The 45° viewpoint of F-35

Figure B.10. Four different viewpoints of the F-35 Lightning fighter model.

B.5.3 B-2 Spirit Bomber RCS and Rmax



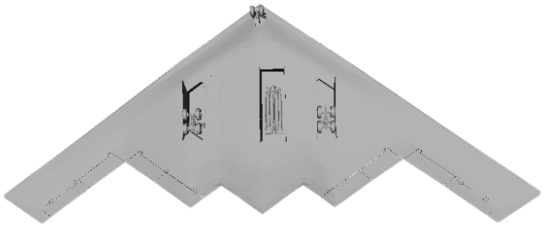
(a) The RCS of B-2 (b) The Rmax of The RCS of B-2

Figure B.11. The polar plots of a B-2 Spirit bomber in azimuth angle ($\theta = 90°$). In (a), the RCS of the object computed by CST studio with the signal frequency = 8GHz. In (b), the Rmax of the sensor to detect the target and calculated by the range equation with the RCS.

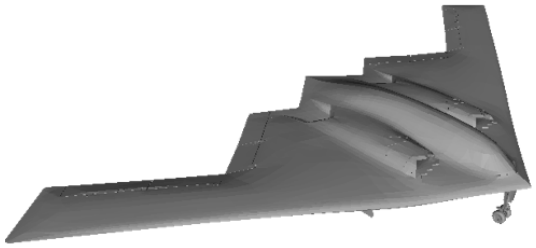
B.5.4 B-2 Spirit Bomber Model in Different Viewpoints

(a) The front viewpoint of B-2

(b) The side viewpoint of B-2



(c) The bottom viewpoint of B-2

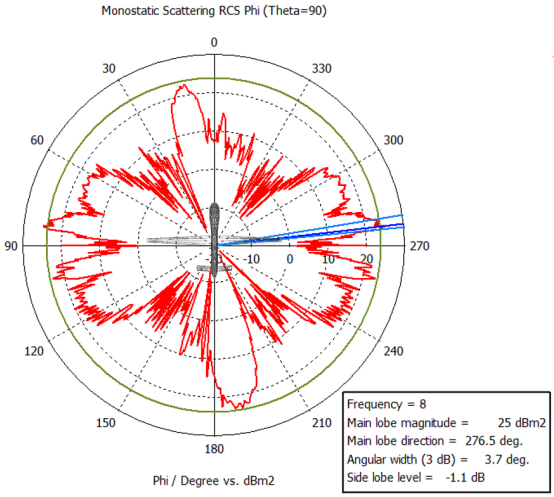


(d) The 45° viewpoint of B-2

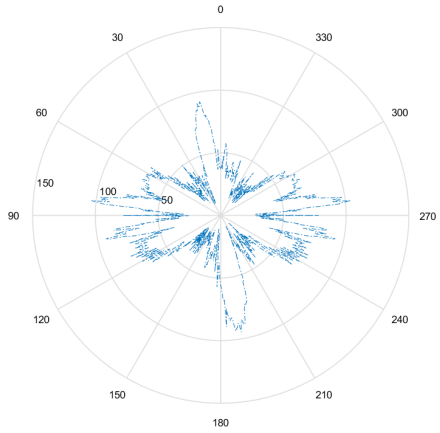
Figure B.12. Four different viewpoints of the B-2 Spirit bomber model.

B.6 UAV

B.6.1 MQ-1 Predator UAV RCS and Rmax



(a) The RCS of MQ-1



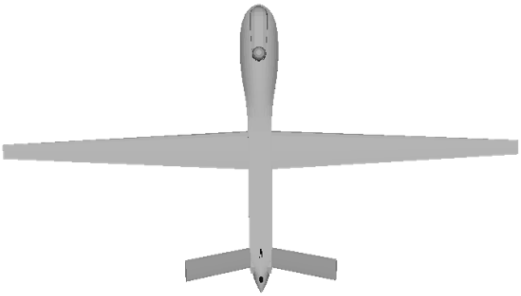
(b) The Rmax of The RCS of MQ-1

Figure B.13. The polar plots of a MQ-1 Predator UAV in azimuth angle (θ = 90°). In (a), the RCS of the object computed by CST studio with the signal frequency = 8GHz. In (b), the Rmax of the sensor to detect the target and calculated by the range equation with the RCS.

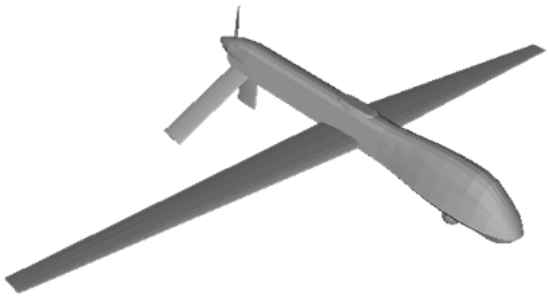
B.6.2 MQ-1 Predator UAV Model in Different Viewpoints

(a) The front viewpoint of MQ-1

(b) The side viewpoint of MQ-1



(c) The bottom viewpoint of MQ-1

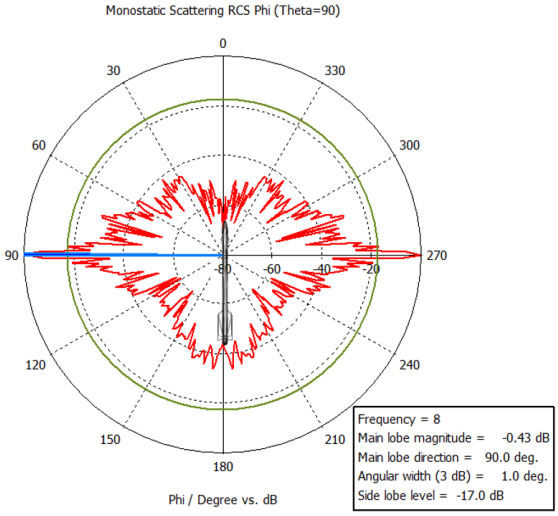


(d) The 45° viewpoint of MQ-1

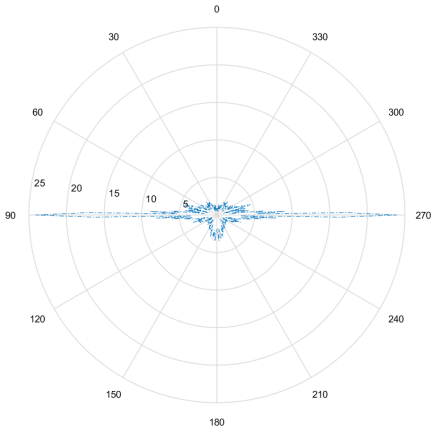
Figure B.14. Four different viewpoints of the MQ-1 Predator UAV model.

B.7 Missiles

B.7.1 AA-11 Archer Air-to-Air Missile RCS and Rmax



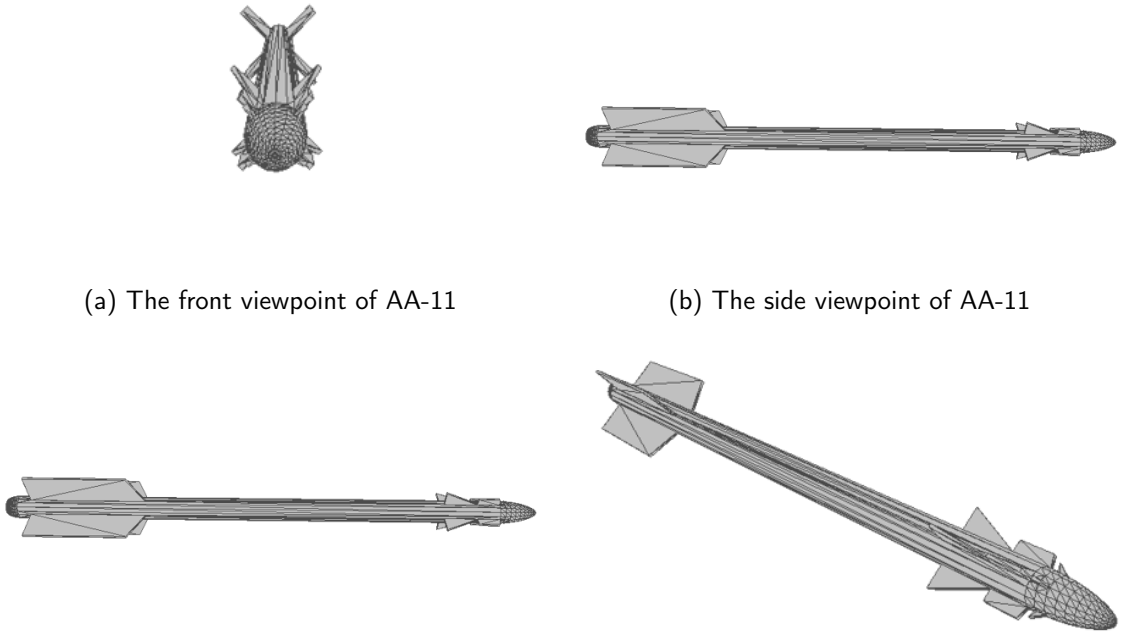
(a) The RCS of MQ-1



(b) The Rmax of The RCS of MQ-1

Figure B.15. The polar plots of a AA-11 Archer air-to-air missile in azimuth angle (θ = 90°). In (a), the RCS of the object computed by CST studio with the signal frequency = 8GHz. In (b), the Rmax of the sensor to detect the target and calculated by the range equation with the RCS.

B.7.2 AA-11 Archer Air-to-Air Missile Model in Different Viewpoints



(a) The front viewpoint of AA-11

(b) The side viewpoint of AA-11

(c) The bottom viewpoint of AA-11

(d) The 45° viewpoint of AA-11

Figure B.16. Four different viewpoints of the AA-11 Archer air-to-air missile model.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C:
Regression Analysis Meta Models

There are several model selection and verification techniques that have been discussed in this work. Each model selection method can result in a meta-model with certain difference than others. These models could be useful for substituting the hybrid sensor model in the simulation scenarios or can be useful for further regression analysis. There is a list of the full predicted expression of proper fit meta-models of predicting TTD and the logistic meta-models for predicting P_d of the sensor. These models are fit with different complexity of predictors or by different model selection techniques that have been introduced in the study. Each mathematical meta-model was produced from simulation replication results using JMP Pro 12.

C.1 TTD Meta-Model Prediction Expression with the Predictors Interaction Terms to Up to Four Degrees (D4)

$$\text{Predicted } TTD = y = (-0.335133972101971) + 0.000109504106388287 *$$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) + -0.0000930384722884577 *$$

$$: Name(\text{"}speed(Km/hr)\text{"}) + 0.0000684307108072084 *$$

$$: Name(\text{"}SweepingItg(s)\text{"}) + -0.00260890738421434 * : thetaAz + 0.0000010249647684912$$

$$* : prf + (: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * ((: Name(\text{"}SweepingItg(s)\text{"}) - 4.53622619566796) * 0.0000364559651439359) + (: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * ((: thetaAz - 4.51469690361972) * 0.000232733268884828) + (: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * ((: prf - 954.780636720455) * -0.0000000614158569719) + (: Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * ((: Name(\text{"}SweepingItg(s)\text{"})$$

$$
\begin{aligned}
&-4.53622619566796) * -0.0000017715916845948) + (:\ Name(\text{"}speed(Km/hr)\text{"})\\
&-689.113631341769) * ((:\ thetaAz - 4.51469690361972) * -0.0000003873677926721) + (\\
&:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * ((:\ prf - 954.780636720455) *\\
&-0.0000000186636191686) + (:\ Name(\text{"}SweepingItg(s)\text{"}) - 4.53622619566796) * ((\\
&:\ thetaAz - 4.51469690361972) * 0.00038657629526604) + (:\ Name(\text{"}SweepingItg(s)\text{"})\\
&-4.53622619566796) * ((:\ prf - 954.780636720455) * 0.0000008795629531869) + (\\
&:\ thetaAz - 4.51469690361972) * ((:\ prf - 954.780636720455) * 0.0000009787295997473)\\
&+ (:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * ((:\ Name(\text{"}SweepingItg(s)\text{"})\\
&-4.53622619566796) * ((:\ thetaAz - 4.51469690361972) *\\
&0.0000003866060444694)) + (:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) *\\
&((:\ Name(\text{"}SweepingItg(s)\text{"}) - 4.53622619566796) * ((:\ prf - 954.780636720455) *\\
&0.0000000888161807738)) + (:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) *\\
&((:\ thetaAz - 4.51469690361972) * ((:\ prf - 954.780636720455) * 0.0000003420567670487)\\
&) + (:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * ((:\ Name(\text{"}SweepingItg(s)\text{"})\\
&-4.53622619566796) * ((:\ thetaAz - 4.51469690361972) * -0.0000005324383684 78)) + (\\
&:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * ((:\ Name(\text{"}SweepingItg(s)\text{"})\\
&-4.53622619566796) * ((:\ prf - 954.780636720455) * 0.0000000061416000062)) + (\\
&:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * ((:\ thetaAz - 4.51469690361972) * ((\\
&:\ prf - 954.780636720455) * 0.0000000043878706677)) + (:\ Name(\text{"}SweepingItg(s)\text{"})\\
&-4.53622619566796) * ((:\ thetaAz - 4.51469690361972) * ((:\ prf - 954.780636720455) *\\
&-0.0000002539686397348)) + (:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534)\\
&* ((:\ Name(\text{"}SweepingItg(s)\text{"}) - 4.53622619566796) * ((:\ thetaAz - 4.51469690361972)\\
&* ((:\ prf - 954.780636720455) * 0.0000004124575542816))) + (:\ Name(\text{"}speed(Km/hr)\text{"})
\end{aligned}
$$

$$-689.113631341769) * ((:\ Name(\text{"}SweepingItg(s)\text{"}) - 4.53622619566796) * ((:\ thetaAz$$

$$-4.51469690361972) * ((:\ prf - 954.780636720455) * -0.00000000407718044))) + ($$

$$:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) *$$

$$((:\ Name(\text{"}IncidentAngle(degree)\text{"})$$

$$-0.349461735240534) * 0.000386865976507422) + ($$

$$:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * (($$

$$:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * (($$

$$:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * -0.000004020545329974)) + ($$

$$:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * (($$

$$:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * (($$

$$:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * (($$

$$:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.349461735240534) * -0.0000035220284762869)))$$

$$+ (:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * ((:\ Name(\text{"}speed(Km/hr)\text{"})$$

$$-689.113631341769) * 0.0000009944186840177) + (:\ Name(\text{"}speed(Km/hr)\text{"})$$

$$-689.113631341769) * ((:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * (($$

$$:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * -2.28060205858768e - 10)) + ($$

$$:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * ((:\ Name(\text{"}speed(Km/hr)\text{"})$$

$$-689.113631341769) * ((:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * (($$

$$:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341769) * 3.33032620551949e - 13))) + (:\ thetaAz$$

$$-4.51469690361972) * ((:\ thetaAz - 4.51469690361972) * 0.00939036904391173) + ($$

$$:\ thetaAz - 4.51469690361972) * ((:\ thetaAz - 4.51469690361972) * ((:\ thetaAz$$

$$-4.51469690361972) * 0.00125308646280733)) + (:\ thetaAz - 4.51469690361972) * (($$

$$:\ thetaAz - 4.51469690361972) * ((:\ thetaAz - 4.51469690361972) * ((:\ thetaAz$$

$-4.51469690361972) * -0.00424325293238701)))$

C.2 TTD Meta-Model Prediction Expression Fit by The Predictor Waypoint_X

Predicted $TTD = y = 0.132755047311965 + -0.0000818679958353739*$

$: Name("speed(Km/hr)") + -0.0041921314510958*$

$: Name("thetaAz(Degree)") + (: Waypoint_X - 0.607415321994461$

$) * (: Name("thetaAz(Degree)") - 4.51469690361972) * 0.000175155053986711 + ($

$: Name("speed(Km/hr)") - 689.113631341766) * (: Name("SweepingInterval(s)")$

$-4.53622619566797) * -0.0000012486135818654 + (: Name("speed(Km/hr)")$

$-689.113631341766) * (: Name("prf(Hz)") - 954.780636720453)*$

$-0.0000000179586873246 + (: Waypoint_X - 0.607415321994461) * ($

$: Name("SweepingInterval(s)") - 4.53622619566797) * (: Name("prf(Hz)")$

$-954.780636720453) * 0.0000000951097193237 + (: Name("speed(Km/hr)")$

$-689.113631341766) * (: Name("SweepingInterval(s)") - 4.53622619566797) * ($

$: Name("prf(Hz)") - 954.780636720453) * 0.0000000091934435226 + ($

$: Name("speed(Km/hr)") - 689.113631341766) * (: Name("prf(Hz)")$

$-954.780636720453) * (: Name("thetaAz(Degree)") - 4.51469690361972)*$

$0.0000000101162589086 + (: Waypoint_X - 0.607415321994461) * ($

$: Name("SweepingInterval(s)") - 4.53622619566797) * (: Name("prf(Hz)")$

$-954.780636720453) * (: Name("thetaAz(Degree)") - 4.51469690361972)*$

$0.0000001444237033903 + (: Name("speed(Km/hr)") - 689.113631341766) * ($

$: Name("SweepingInterval(s)") - 4.53622619566797) * (: Name("prf(Hz)")$

$$-954.780636720453) * (:\ Name(\text{"}thetaAz(Degree)\text{"}) - 4.51469690361972) *$$
$$-0.0000000067851953957 + (:\ Waypoint_X - 0.607415321994461) * (:\ Waypoint_X$$
$$-0.607415321994461) * 0.00013212709789754 + (:\ Waypoint_X - 0.607415321994461) * ($$
$$:\ Waypoint_X - 0.607415321994461) * (:\ Waypoint_X - 0.607415321994461) *$$
$$-0.0000006358281245164 + (:\ Waypoint_X - 0.607415321994461) * (:\ Waypoint_X$$
$$-0.607415321994461) * (:\ Waypoint_X - 0.607415321994461) * (:\ Waypoint_X$$
$$-0.607415321994461) * -0.0000003832223284612 + (:\ Name(\text{"}speed(Km/hr)\text{"})$$
$$-689.113631341766) * (:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341766) *$$
$$0.0000000975438480005 + (:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341766) * ($$
$$:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341766) * (:\ Name(\text{"}speed(Km/hr)\text{"})$$
$$-689.113631341766) * -0.0000000003982077417 + (:\ Name(\text{"}speed(Km/hr)\text{"})$$
$$-689.113631341766) * (:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341766) * ($$
$$:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.113631341766) * (:\ Name(\text{"}speed(Km/hr)\text{"})$$
$$-689.113631341766) * 6.75431897182834e - 13 + (:\ Name(\text{"}thetaAz(Degree)\text{"})$$
$$-4.51469690361972) * (:\ Name(\text{"}thetaAz(Degree)\text{"}) - 4.51469690361972) *$$
$$0.0104815940468138 + (:\ Name(\text{"}thetaAz(Degree)\text{"}) - 4.51469690361972) * ($$
$$:\ Name(\text{"}thetaAz(Degree)\text{"}) - 4.51469690361972) * (:\ Name(\text{"}thetaAz(Degree)\text{"})$$
$$-4.51469690361972) * 0.0022672505711434 + (:\ Name(\text{"}thetaAz(Degree)\text{"})$$
$$-4.51469690361972) * (:\ Name(\text{"}thetaAz(Degree)\text{"}) - 4.51469690361972) * ($$
$$:\ Name(\text{"}thetaAz(Degree)\text{"}) - 4.51469690361972) * (:\ Name(\text{"}thetaAz(Degree)\text{"})$$
$$-4.51469690361972) * -0.0048413478211252$$

C.3 TTD Meta-Model Prediction Expression with CV the Predictors Interaction Terms to Up to Four Degrees (D4)

$$\text{Predicted } TTD = y = 0.120036628542651$$

$$+0.0000983742927557509 * : Name(\text{"}IncidentAngle(degree)\text{"}) +$$

$$-0.0000799496570348101 * : Name(\text{"}speed(Km/hr)\text{"}) + -0.000253545583416296 *$$

$$: Name(\text{"}SweepingItg(s)\text{"}) + -0.0000014682311208055$$

$$* : prf + -0.000403100130069902$$

$$* : thetaAz + (: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * (($$

$$: Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * 0.000000113281426919) + ($$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * (($$

$$: Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * 0.0000173804822553529) + ($$

$$: Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * ((: Name(\text{"}SweepingItg(s)\text{"})$$

$$-4.58864175824178) * 0.0000004732888950157) + (: Name(\text{"}speed(Km/hr)\text{"})$$

$$-689.031102417577) * ((: prf - 950.415604395603) * 0.0000000042296479775) + ($$

$$: Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * ((: prf - 950.415604395603) *$$

$$-0.0000003121969163524) + (: Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * (($$

$$: thetaAz - 4.52115750915751) * -0.00010674854891087) + ($$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * ((: Name(\text{"}speed(Km/hr)\text{"})$$

$$-689.031102417577) * ((: Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) *$$

$$0.0000000519603323764)) + (: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) *$$

$$((: Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * ((: thetaAz - 4.52115750915751) *$$

$$0.0000001766180179292)) + (: Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * (($$

$$
\begin{aligned}
&(:\ Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * ((:\ prf - 950.415604395603) * \\
&0.0000000023050224958)) + (:\ Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * ((:\ prf \\
&- 950.415604395603) * ((:\ thetaAz - 4.52115750915751) * - 0.0000035189118965 93)) + (\\
&:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * ((:\ Name(\text{"}speed(Km/hr)\text{"}) \\
&- 689.031102417577) * ((:\ Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * ((:\ prf \\
&- 950.415604395603) * 0.0000000003224874102))) + (:\ Name(\text{"}IncidentAngle(degree)\text{"}) \\
&- 0.236844281196993) * ((:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * ((\\
&:\ Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * ((:\ thetaAz - 4.52115750915751) * \\
&0.000000063925169957))) + (:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * ((\\
&:\ Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * ((:\ prf - 950.415604395603) * ((\\
&:\ thetaAz - 4.52115750915751) * 0.0000000015225511188))) + (\\
&:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * ((\\
&:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * 0.000306793345669167) + (\\
&:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * ((\\
&:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * ((\\
&:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * - 0.0000019714334301779)) \\
&+ (:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * ((\\
&:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * ((\\
&:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * ((\\
&:\ Name(\text{"}IncidentAngle(degree)\text{"}) - 0.236844281196993) * - 0.0000027142926267193))) \\
&+ (:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * ((:\ Name(\text{"}speed(Km/hr)\text{"}) \\
&- 689.031102417577) * 0.0000000946872268113) + (:\ Name(\text{"}speed(Km/hr)\text{"}) \\
&- 689.031102417577) * ((:\ Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * ((
\end{aligned}
$$

$$: Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * -0.0000000004149640484)) + ($$

$$: Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * ((: Name(\text{"}speed(Km/hr)\text{"})$$

$$-689.031102417577) * ((: Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * ((: Name(\text{"}speed(Km/hr)\text{"}) - 689.031102417577) * 7.04714557562147e - 13))) + ($$

$$: Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * ((: Name(\text{"}SweepingItg(s)\text{"})$$

$$-4.58864175824178) * ((: Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) *$$

$$0.00002354671861772 47)) + (: Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * ((: Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) * ((: Name(\text{"}SweepingItg(s)\text{"})$$

$$-4.58864175824178) * ((: Name(\text{"}SweepingItg(s)\text{"}) - 4.58864175824178) *$$

$$0.0000110131651500396))) + (: prf - 950.415604395603) * ((: prf - 950.415604395603) *$$

$$0.0000000098995110322) + (: prf - 950.415604395603) * ((: prf - 950.415604395603) * ((: prf - 950.415604395603) * -8.96320574789512e - 12)) + (: prf - 950.415604395603) * ((: prf - 950.415604395603) * ((: prf - 950.415604395603) * ((: prf - 950.415604395603)$$

$$* - 3.60832441034479e - 14))) + (: thetaAz - 4.52115750915751) * ((: thetaAz$$

$$-4.52115750915751) * 0.000270040481090055)$$

C.4 P_d Logistic Meta-Model Prediction Expression with the Predictors Interaction Terms to Up to Two Degrees (D2)

$$logit(Y) = 5.49290082155754 + 0.0152221773190791 * : Name(\text{"}IncidentAngle(degree)\text{"}) +$$

$$-0.00287798318177724 * : Name(\text{"}speed(Km/hr)\text{"}) + 0.255867723581157 *$$

$$: Name(\text{"}SweepingItg(s)\text{"}) + 0.00280016312058323 * : prf + 0.599658190605475 *$$

$$: thetaAz + (: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.104020677342242) * ($$

$$: Name(\text{"}speed(Km/hr)\text{"}) - 699.234986458056) * -0.000102076880645261 + ($$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.104020677342242) * (: prf - 937.285374710625)$$

$$* 0.0000500667665823787 + (: Name(\text{"}speed(Km/hr)\text{"}) - 699.234986458056) * ($$

$$: Name(\text{"}SweepingItg(s)\text{"}) - 4.49523687037571) * -0.000833411712584499 + ($$

$$: Name(\text{"}speed(Km/hr)\text{"}) - 699.234986458056) * (: thetaAz - 4.50015433266254) *$$

$$-0.000530166324573227 + (: prf - 937.285374710625) * (: thetaAz - 4.50015433266254) *$$

$$0.000900239016858853 + (: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.104020677342242) * ($$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) - 0.104020677342242) * -0.115718733645825 + ($$

$$: Name(\text{"}SweepingItg(s)\text{"}) - 4.49523687037571) * (: Name(\text{"}SweepingItg(s)\text{"})$$

$$-4.49523687037571) * -0.0774609407029976 + (: prf - 937.285374710625) * (: prf$$

$$-937.285374710625) * -0.0000044260381675404 + (: thetaAz - 4.50015433266254) * ($$

$$: thetaAz - 4.50015433266254) * 0.204106328035467$$

$$Probability(NoDetect = 1) = \frac{1}{1 + e^{-logit(Y)}}$$

$$Probability(Detect = 1) = \frac{1}{1 + e^{logit(Y)}}$$

C.5 P_d Logistic Meta-Model Prediction Expression with the Predictors Interaction Terms to Up to Four Degrees (D4)

$$logit(Y) = 6.19138673585242 + -0.00251757832129637 * : Name(\text{"}speed(Km/hr)\text{"})$$

$$+0.298714639447719 * : Name(\text{"}SweepingItg(s)\text{"})$$

$$+0.00275270585578442 * : prf$$

$$+0.527282207665547 * : thetaAz + : Name(\text{"}IncidentAngle(degree)\text{"}) * (($$

$$: Name(\text{"}speed(Km/hr)\text{"}) - 700.000626702996) * -0.000047871531763866) +$$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) * ((: prf - 937.500363306097) *$$

$$-0.0000481640975487524) + (: Name(\text{"}speed(Km/hr)\text{"})$$

$$-700.000626702996) * ((: Name(\text{"}SweepingItg(s)\text{"}) - 4.50045413260674) * -0.000450167831376932) +$$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) * ((: Name(\text{"}speed(Km/hr)\text{"})$$

$$-700.000626702996) *$$

$$((: Name(\text{"}SweepingItg(s)\text{"}) - 4.50045413260674) * 0.0000210328374017812)) +$$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) * (: Name(\text{"}IncidentAngle(degree)\text{"}) *$$

$$-0.136342324810703) + : Name(\text{"}IncidentAngle(degree)\text{"}) * ($$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) * (: Name(\text{"}IncidentAngle(degree)\text{"}) *$$

$$0.000433908416509389)) + : Name(\text{"}IncidentAngle(degree)\text{"}) * ($$

$$: Name(\text{"}IncidentAngle(degree)\text{"})$$

$$*(: Name(\text{"}IncidentAngle(degree)\text{"}) * ($$

$$: Name(\text{"}IncidentAngle(degree)\text{"}) * 0.000115447885283602))) + ($$

$$: Name(\text{"}SweepingItg(s)\text{"}) - 4.50045413260674) * ((: Name(\text{"}SweepingItg(s)\text{"})$$

$$-4.50045413260674) * -0.0534218264810987) + (: prf - 937.500363306097) * ((: prf$$

$$-937.500363306097) * ((: prf - 937.500363306097) * ((: prf - 937.500363306097) *$$

$$-9.38489742883684e - 12)))$$

In general, once the $logit(Y)$ is computed, the probability of no detection of the target by this DoE is calculated by:

$$\text{Probability(NoDetect=1)} = \frac{1}{1 + e^{-logit(Y)}} \tag{C.1}$$

The probability of detection of the target by this DoE is calculated by:

$$\text{Probability(NoDetect=0)} = 1 - \text{Probability(NoDetect=1)} = \frac{1}{1 + e^{logit(Y)}} \qquad \text{(C.2)}$$

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] A. H. Buss, private communication, Novewmber 2016.

[2] T. W. Lucas, "The stochastic versus deterministic argument for combat simulations: Tales of when the average won't do," *Military Operations Research*, vol. 5, no. 3, pp. 9–28, 2000.

[3] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the Conference on Winter Simulation*, 2005, pp. 130–143.

[4] M&S VV&A RPG core document: Introduction. (2011, Jan.). Modeling & Simulation Coordination Office. [Online]. Available: http://www.msco.mil/documents/Key01_Key_Concepts.pdf

[5] J. P. Powers and P. E. Pace, "An actively mode-locked fiber laser for sampling in a wide-bandwidth opto-electronic analog-to-digital converter," in *Lasers and Applications in Science and Engineering*, 2008, pp. 687 322–687 322.

[6] Y. W. Tan, C. H. Nam, and P. E. Pace, "Effects of amplitude and timing jitter on the performance of photonic sigma-delta modulators," in *Proc. of SPIE Vol*, 2011, vol. 7941, pp. 79 411B–1.

[7] Y. P. Cheng, "Detection of frequency hopped signals timing information using the temporal correlation function," M.S. thesis, Electrical and Computer Engineering Department, Naval Postgraduate School, California, USA, Sep. 2008.

[8] M. Misiti, Y. Misiti, G. Oppenheim, and J.-M. Poggi. (1996, Mar.). Wavelet toolbox for use with MATLAB user's guide. The MathWorks, Inc. [Online]. Available: http://web.mit.edu/1.130/WebDocs/wavelet_ug.pdf

[9] P. E. Pace, *Detecting and Classifying Low Probability of Intercept Radar*, 2nd ed. Norwood, MA: Artech house, 2009, ch. 14, p. 481.

[10] P. E. Pace, *Detecting and Classifying Low Probability of Intercept Radar*, 2nd ed. Norwood, MA: Artech house, 2009, ch. 14, p. 483.

[11] J. A. Dewar, J. Gillogly, and M. L. Juncosa, *Non-monotonicity, chaos, and combat models*. Santa Monica, CA: Rand, 1991.

[12] B. Bennett, J. Hagen, J. Chow, A. Brooks, P. Myrick, W. Hobbs, and R. Gates, "Alternatives for enhancing aircraft survivability," *RAND*, vol. 1, no. MR-1003, 1998.

[13] W. Lei, C. Ming-yan, Z. Wei, and L. Xian-liang, "A universal program framework for radar system simulation," in *Computer Science and Automation Engineering (CSAE), IEEE International Conference on*, 2012, vol. 1, pp. 595 – 599.

[14] S. V. Migel and I. G. Prokopenko, "Radar simulation system managed by script language," in *International Radar Symposium (IRS)*, 2014, pp. 1–5.

[15] A. H. Buss and P. J. Sanchez, "Simple movement and detection in discrete event simulation," in *Proceedings of the Winter Simulation Conference*, 2005, p. 9.

[16] M. R. Inggs, C. A. Tong, A. K. Mishra, and F. D. V. Maasdorp, "Modelling and simulation in commercial radar system design," in *Radar Systems, IET International Conference on*, 2012, pp. 1–5.

[17] W. Quanmin, W. Chuncai, G. Gang, and H. Kedi, "Rf effect algorithms of terrain environment in signal-level radar system simulation," in *Computer Modeling and Simulation, ICCMS'10. Second International Conference on*, 2010, vol. 4, pp. 178 –181.

[18] T. Cioppa and T. Lucas, "Efficient nearly orthogonal and space-filling latin hypercubes," *Technometrics*, vol. 49, no. 1, p. 45, feb 2007.

[19] D. Adams. (n.d.). BrainyQuote. [Online]. Available: https://www.brainyquote.com/quotes/quotes/d/douglasada161654.html. Accessed Dec. 7, 2016.

[20] Y. P. Cheng and D. Brutzman, "Matlab and simulink creation and animation of x3d in web-based simulation," in *Proceedings of the International Conference on 3D Web Technology*, 2015, pp. 169–169.

[21] W. L. Stutzman and G. A. Thiele, *Antenna Theory and Design*, 2nd ed. San Francisco, CA: John Wiley & Sons, Inc., 1998, ch. 3, pp. 87, 121–122.

[22] D. Brutzman and L. Daly, *X3D, Extensible 3D Graphics for Web Authors*. Burlington, MA: Morgan Kaufmann, 2007, ch. 1, p. xix.

[23] D. Jenn, "Arrays," class notes for Antennas and Propagation, Dept. of Electrical and Computer Engineering Department, Naval Postgraduate School, Monterey, CA, summer 2013.

[24] *Virtual Reality Toolbox For Use with MATLAB and Simulink*, 4th ed., HUMUSOFT s.r.o. and The MathWorks, Inc., Natick, MA, 2004, pp. 128–130. Available: http://cda.psych.uiuc.edu/matlab_pdf/vr.pdf

[25] Free fall with examples. (n.d.). www.PhysicsTutorials.org. [Online]. Available: http://www.physicstutorials.org/home/mechanics/1d-kinematics/free-fall. Accessed Nov. 9, 2016.

[26] What is Hooke's law? (n.d.). www.khanacademy.org. [Online]. Available: https://www.khanacademy.org/science/physics/work-and-energy/hookes-law/a/what-is-hookes-law. Accessed Nov. 9, 2016.

[27] J. Banks, "Discrete event simulation," in *Proceedings of Winter Simulation Conference*, 1999, pp. 7–13.

[28] L. Schruben, "Simulation modeling with event graphs," *Communications of the ACM*, vol. 26, no. 11, pp. 957–963, 1983.

[29] A. Buss, "Discrete event simulation modeling," class notes for Introduction to Discrete Event Simulation Modeling, Naval Postgraduate School, 2013 spring. Available: https://eos.nps.edu/Simkit/docs/Discrete%20Event%20Simulation%20Modeling.docx

[30] A. H. Buss and P. J. Sanchez, "Modeling very large scale systems: Building complex models with legos (listener event graph objects)," in *Proceedings of the Conference on Winter Simulation: Exploring New Frontiers*, 2002, pp. 732–737.

[31] A. Alrowaie, "The effect of time-advance mechanism in modeling and simulation," Ph.D. dissertation, MOVES Institute, Naval Postgraduate School, Monterey, CA, Sep. 2011. Available: https://calhoun.nps.edu/handle/10945/10798

[32] A. Buss and A. Al Rowaei, "A comparison of the accuracy of discrete event and discrete time," in *Proceedings of the Winter Simulation Conference*, 2010, pp. 1468–1477.

[33] J. Nutaro, "Constructing multi-point discrete event integration schemes," in *Proceedings of the conference on Winter simulation*, 2005, pp. 267–273.

[34] K. Y. Lin, class notes for Stochastic Models I, Dept. of Operation Research, Naval Postgraduate School, Monterey, CA, summer 2014.

[35] N. Levanon, *Radar Principles*. Hoboken, NJ: A Wiley-Interscience Publication, 1988, ch. 1, pp. 1–13.

[36] P. E. Pace, *Detecting and Classifying Low Probability of Intercept Radar*, 2nd ed. Norwood, MA: Artech house, 2009, ch. 1, pp. 26–27.

[37] M. Skolnik, *Introduction to Radar Systems*, 3rd ed. New York, NY: McGrawHill, 2002, ch. 2, pp. 30–31.

[38] B. R. Mahafza, *Radar Systems Analysis and Design using Matlab*, 3rd ed. Boca Raton, FL: CRC Press, 2012, ch. 14, pp. 485–487.

[39] M. Skolnik, *Introduction to Radar Systems*, 3rd ed. New York, NY: McGrawHill, 2002, ch. 2, pp. 49–54.

[40] M. Skolnik, *Introduction to Radar Systems*, 3rd ed. New York, NY: McGrawHill, 2002, ch. 2, p. 57.

[41] M. Skolnik, *Introduction to Radar Systems*, 3rd ed. New York, NY: McGrawHill, 2002, ch. 2, p. 55.

[42] D. Q. Nykamp. (n.d.). Spherical coordinates. Math Insight. [Online]. Available: http://mathinsight.org/spherical_coordinates. Accessed Nov. 9, 2016.

[43] CST microwave studio. (n.d.). CST Computer Simulation Technology. [Online]. Available: https://www.cst.com/Products/CSTMWS. Accessed Nov. 9, 2016.

[44] J. Haferman. (2016, Apr.). High performance computing Hamming cluster architecture. *NPS Wiki*. Naval Postgraduate School. [Online]. Available: https://wiki.nps.edu/display/HPC/Hamming+Architecture

[45] M. Skolnik, *Introduction to Radar Systems*, 3rd ed. New York, NY: McGrawHill, 2002, ch. 2, pp. 45–49.

[46] B. R. Mahafza, *Radar Systems Analysis and Design using Matlab*, 3rd ed. Boca Raton, FL: CRC Press, 2012, ch. 13, p. 438.

[47] M. Skolnik, *Introduction to Radar System*, 3rd ed. New York, NY: McGrawHill, 2002, ch. 2, pp. 66–73.

[48] O. Pescetti. (n.d.). Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Perfect_is_the_enemy_of_good. Accessed Dec. 7, 2016.

[49] D. D. Wackerly, W. I. Mendenhall, and R. L. Scheaffer, *Mathematical Statistics with Applications*, 7th ed. Belmont, CA: Brooks Cole/Cengage Learning, 2008, ch. 4, p. 185.

[50] Andale. (2014, Jan.). Gamma distribution: Definition, finding in excel, MATLAB. [Online]. Available: http://www.statisticshowto.com/gamma-distribution/

[51] M. Fowler, *UML distilled: a brief guide to the standard object modeling language*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley Professional, 2000, ch. 5, pp. 68–70.

[52] D. Diderot. (n.d.). BrainyQuote. [Online]. Available: https://www.brainyquote.com/quotes/quotes/d/denisdider393574.html. Accessed Dec. 7, 2016.

[53] A. M. Law, *Simulation Modeling and Analysis*, 5th ed. New York, NY: McGrawHill Education, 2013, ch. 12, p. 629.

[54] A. M. Law, *Simulation Modeling and Analysis*, 5th ed. New York, NY: McGrawHill Education, 2013, ch. 12, pp. 632–633.

[55] A. M. Law, *Simulation Modeling and Analysis*, 5th ed. New York, NY: McGrawHill Education, 2013, ch. 12, pp. 671–672.

[56] S. M. Sanchez, T. W. Lucas, P. J. Sanchez, C. J. Nannini, and H. Wan, "Designs for large-scale simulation experiments, with applications to defense and homeland security," Faculty and Researcher Publications Collection, Operation Research Department, Naval Postgraduate School, California, USA, 2012.

[57] A. H. Buss. (n.d.). Using simkit in Netbeans and Eclipse. Naval Postgraduate School. [Online]. Available: https://eos.nps.edu/Simkit/. Accessed Nov. 1, 2016.

[58] R. Hamming. (n.d.). BrainyQuote. [Online]. Available: https://www.brainyquote.com/quotes/quotes/r/richardham645682.html. Accessed Dec. 7, 2016.

[59] J. L. Devore, *Probability and Statistics for Engineering and The Sciences*, 6th ed. Belmont, CA: Brooks/Cole Thomson Learning, 2004, ch. 12, pp. 496–516.

[60] A. Tallavajhula and A. Kelly, "Construction and validation of a high fidelity simulator for a planar range sensor," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6261 – 6266.

[61] S. Chatterjee, A. S. Hadi, and B. Price, *Regression Analysis by Example*, 3rd ed. Hoboken, NJ: Wiley-Interscience, 2000, ch. 1, pp. 1–18.

[62] R. Koyak, "One-way analysis of variance (anova)," class notes for Data Analysis, Monterey, CA, spring 2016.

[63] R. Koyak, "Linear models and estimation," class notes for Data Analysis, Dept. of Operation Research, Naval Postgraduate School, Monterey, CA, spring 2016.

[64] E. W. Weisstein. (n.d.). Least squares fitting. MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/LeastSquaresFitting.html. Accessed Nov. 9, 2016.

[65] R. Koyak, "Unit 6.3 model selection," class notes for Data Analysis, Dept. of Operation Research, Naval Postgraduate School, spring 2016.

[66] J. J. Faraway, *Extending the Linear Model with R, Generalized Linear, MixedEffects and Nonparametric Regression Models*. Boca Raton, FL: CRC Press, 2016, ch. Random Effect, p. 203.

[67] S. W. Menard, *Logistic Regression From Introductory to Advanced Concepts and Applications*. Thousand Oaks, CA: SAGE Publication, 2010, ch. 1, pp. 12–18.

[68] J. W. Osborne, "Sweating the small stuff in educational psychology: how effect size and power reporting failed to change from 1969 to 1999, and what that means for the future of changing practices 1," *Educational Psychology*, vol. 28, no. 2, pp. 151–160, 2008.

[69] J. W. Osborne, "Improving your data transformations: Applying the box-cox transformation," *Practical Assessment, Research & Evaluation*, vol. 15, no. 12, pp. 1–9, 2010.

[70] J. J. Faraway, *Linear Model with R*. Boca Raton, FL: CRC Press, 2015, ch. 9, p. 134.

[71] A. Buthmann. Making data normal using Box-Cox power transformation. Six Sigma.com. Available: https://www.isixsigma.com/tools-templates/normality/making-data-normal-using-box-cox-power-transformation/

[72] Cross-validation. (n.d.). Perception Sensing Instrumentation Lab, Texas A&M University. [Online]. Available: http://research.cs.tamu.edu/prism/lectures/pr/pr_l13.pdf. Accessed Nov. 9, 2016.

[73] R. Koyak, "Model selection," class notes for Data Analysis, Dept. of Operation Research, Naval Postgraduate School, Monterey, CA, spring 2016.

[74] Using validation. (n.d.). jmp SAS online support. [Online]. Available: http://www.jmp.com/support/help/Using_Validation.shtml. Accessed Nov. 9, 2016.

[75] S. Buttery and L. Whitaker, "Model complexity," class notes for Advanced Data Analysis, Dept. of Operation Research, Naval Postgraduate School, Monterey, CA, summer 2016.

[76] K.-E. J. Ang, "Extending orthogonal and nearly orthogonal latin hypercube designs for computer simulation and experiments," M.S. thesis, Operation Research Department, Naval Postgraduate School, California, USA, Dec. 2006.

[77] A. B. U. o. S. JMP, *JMP Statistics and Graphics Guid*, 7th ed. SAS Institute Inc., Cary, NC, USA, may 2007, ch. 28, pp. 571–574.

[78] K. Markham. (2014, Mar.). Simple guide to confusion matrix terminology. Data school. [Online]. Available: http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

[79] H. Abdi, "O'Brien test for homogeneity of variance," *Encyclopedia of Measurement and Statistics*, vol. 2, pp. 701–704, 2007.

[80] G. E. P. Box. (n.d.). [Online]. Available: https://www.goodreads.com/quotes/7729857-essentially-all-models-are-wrong-but-some-are-useful. Accessed Dec. 7, 2016.

[81] Y. P. Cheng, P. E. Pace, D. Brutzman, and A. H. Buss, "Hybrid high-fidelity modeling of radar scenarios using atemporal, discrete event, and time-step simulation," in *Proceedings of the World Congress on Engineering and Computer Science*, 2016, vol. 2, pp. 764–770.

[82] D. Brutzman. (n.d.). Savage developers guide. MOVES Institute, Naval Postgraduate School. [Online]. Available: https://savage.nps.edu/Savage/developers.html#DES. Accessed Nov. 9, 2016.

[83] J. Danek. (n.d.). vrml.m function for *.fig to *.wrl conversion. HUMUSOFT. HUMUSOFT. Available: http://www.mathworks.com/matlabcentral/fileexchange/48242-vrml-h--filename--opts-/content/vrml.m. Accessed Nov. 9, 2014.

[84] D.-J. Kroon. (2011, Jul.). Matlab 3D figure to 3D (X)HTML. The MathWorks, Inc. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/32207-matlab-3d-figure-to-3d--x-html

[85] Simulink 3D animation example: Bouncing ball. (n.d.). The MathWorks, Inc. [Online]. Available: http://www.mathworks.com/help/sl3d/examples/bouncing-ball.html. Accessed Nov. 9, 2016.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California